НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2020 р.

Дипломний проєкт

на здобуття ступеня бакалавра

за освітньо-професійною програмою «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

спеціальності 121 Інженерія програмного забезпечення

на тему: «Програмна платформа для обліку волонтерської роботи»

Виконав:
студент IV курсу, групи КП-62
Маслов Нікіта Сергійович _____

Керівник:
Старший викладач кафедри ПЗКС, к.т.н.,
Гречко Анастасія Валеріївна _____

Консультант з нормоконтролю:
Доцент кафедри ПЗКС, к.т.н., доцент,
Онай Микола Володимирович _____

Рецензент:
Доцент кафедри СПСКС, к.т.н, доцент,
Бояринова Юлія Євгеніївна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2020 року

Національний технічний університет України

«Київський політехнічний інститут імені Ігоря Сікорського»

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення комп'ютерних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Науковий керівник кафедри

_____ Іван ДИЧКА

«___»_____2019 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Маслову Нікіті Сергійовичу

1. Тема проєкту «Програмна платформа для обліку волонтерської роботи», керівник проекту ст. викладач Гречко Анастасія Валеріївна, затверджені наказом по університету від «25» травня 2020 р. №1181-с

2. Термін подання студентом проєкту «11» червня 2020 р.

3. Вихідні дані до проєкту: див. Технічне завдання.

4. Зміст пояснювальної записки:
   - аналіз предметної області та існуючих рішень;
   - аналіз технологій розроблення, мов програмування та способів побудови програмних застосувань;
   - опис розроблених програмних засобів;
   - аналіз реалізації програмного забезпечення.

5. Перелік обов'язкового графічного матеріалу:
   - компоненти платформи (креслення);
   - структура бази даних (креслення);

- графічне компонент відображення статистики (плакат);
- структура платформи (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2019 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	14.11.2019	
2.	Розроблення та узгодження технічного завдання	28.11.2019	
3.	Розроблення структури web-ресурсу	15.12.2019	
4.	Підготовка матеріалів першого розділу дипломного проєкту	30.12.2019	
5.	Розроблення дизайну сторінок та графічних елементів	03.02.2020	
6.	Підготовка матеріалів другого розділу дипломного проєкту	20.02.2020	
7.	Програмна реалізація web-ресурсу	10.03.2020	
8.	Тестування web-ресурсу	17.03.2020	
9.	Підготовка матеріалів третього розділу дипломного проєкту	30.03.2020	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	11.04.2020	
11.	Підготовка графічної частини дипломного проєкту	21.04.2020	
12.	Оформлення документації дипломного проєкту	26.05.2020	

Студент Нікіта МАСЛОВ

Керівник проєкту Анастасія ГРЕЧКО

АНОТАЦІЯ

Даний дипломний проект присвячений створенню програмної системи обліку волонтерської роботи учнів та студентів.

Проведено докладний аналіз наявних аналогів, що ведуть облік волонтерських робіт. Було проаналізовано необхідні функціональні та нефункціональні вимоги до розроблюваного продукту, виділено необхідні можливості, що має надавати система своїм користувачам.

Розроблена програмна система є веб-додатком, що містить динамічно оновлювані сторінки, призначені для роботи директорів з подіями та учнями і студентами, мобільний додаток для роботі учнів з подіями, та сервер для взаємодії з базою даних. У системі є два види користувачів - директори навчальних закладів та учні і студенти, які беруть участь у волонтерських подіях. Директори мають доступ до веб-додатку та можуть створювати події, додавати нових учнів та змінювати статус відміток кожного свого учня. Учні мають доступ до мобільного додатку та можуть приймати участь у подіях та створювати відмітки про участь у волонтерських подіях.

У даному дипломному проекті розроблено: архітектуру web-додатку, архітектуру мобільного додатку, алгоритм роботі з подіями за допомогою веб-додатку та мобільного додатку, алгоритм аналізу взаємодії волонтерів з подіями, а також реалізовано графічні елементи та адаптивний дизайн web-сторінок та сторінок у мобільному додатку.

ABSTRACT

This diploma project is devoted to the creation of a software system for accounting for volunteer work of pupils and students.

A detailed analysis of the available analogues that keep records of volunteer work. The necessary functional and non-functional requirements to the developed product were analyzed, the necessary opportunities that the system should provide to its users were identified.

The developed software system is a web application that contains dynamically updated pages designed for principals to work with events and students, a mobile application for students to work with events, and a server for interacting with the database. The system has two types of users - school principals and pupils and students who participate in volunteer events. Principals have access to a web application and can create events, add new students, and change the status of each of their students' grades. Students have access to a mobile app and can take part in events and create notes about participating in volunteer events.

In this diploma project developed: web-application architecture, mobile application architecture, algorithm for working with events using web application and mobile application, algorithm for analyzing the interaction of volunteers with events, as well as implemented graphic elements and adaptive design of web-pages and pages in mobile application.

ДП.045440-01-90 Програмна платформа для обліку волонтерської роботи. Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Платформа для обліку	5	
	волонтерської роботи.		
	Технічне завдання		
ДП.045440-03-81	Платформа для обліку	50	
	волонтерської роботи.		
	Пояснювальна записка		
ДП.045440-04-51	Платформа для обліку	4	
	волонтерської роботи.		
	Програма та методика		
	тестування		
ДП.045440-05-34	Платформа для обліку	14	
	волонтерської роботи.		
	Керівництво користувача		
ДП.045440-06-99	Платформа для обліку	1	
	волонтерської роботи.		
	Архітектура системи.		
	Компоненти платформи		
ДП.045440-07-99	Платформа для обліку	1	
	волонтерської роботи.		
	Схема бази даних.		
	ER-діаграма		
ДП.045440-08-98	Платформа для обліку	1	
	волонтерської роботи.		
	Компакт-диск		

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»
Науковий керівник кафедри
_____ Іван ДИЧКА
«___» _____ 2019 р.

ПРОГРАМНА ПЛАТФОРМА ДЛЯ ОБІКУ ВОЛОНТЕРСЬКОЇ РОБОТИ

Технічне завдання

ДП.045440-02-91

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Анастасія ГРЕЧКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Нікіта МАСЛОВ

2019

ЗМІСТ

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Платформа для обліку волонтерської роботи.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка призначена для використання в якості інформаційне забезпечення для облегшення та покращення процесу обліку волонтерських робіт учнів навчальних закладів.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Платформа повинена забезпечувати такі основні функції:

1) можливість динамічного оновлення вмісту web-сторінок;
2) авторизований доступ до платформи;
3) можливість створювати, редагувати та видаляти події;
4) можливість директорам отримувати статистику по школі;
5) підтримка веб-додатку та мобільного додатку;
6) дозволяє волонтерам створювати відмітки про участь у волонтерських роботах.

Розробку виконати на платформі Node.js з використанням технології AJAX.

Додаткові вимоги:

1) наявність динамічного меню;
2) наявність анімованих кнопок;

3) адаптивний дизайн для веб-додатку;

4) дизайн сторінок з використанням в якості базових білого та синього кольорів

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

1) пояснювальна записка;

2) програма та методика тестування;

3) керівництво користувача;

4) креслення:

- «Компоненти платформи»;
- «Структура бази даних».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи……..14.11.2019

Розроблення та узгодження технічного завдання……....28.11.2019

Розроблення структури платформи..…….....15.12.2019

Розроблення дизайну сторінок та графічних елементів……..03.02.2020

Програмна реалізація платфформи……....17.03.2020

Тестування платформи...…….....03.04.2020

Підготовка матеріалів текстової частини проєкту……..….28.04.2020

Підготовка матеріалів графічної частини проєкту……..…12.05.2020

Оформлення технічної документації проєкту……..…25.05.2020

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до “Програми та методики тестування”.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2020 р.

ПРОГРАМНА ПЛАТФОРМА ДЛЯ ОБЛІКУ ВОЛОНТЕРСЬКОЇ РОБОТИ

Пояснювальна записка

ДП.045440–03–81

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Анастасія ГРЕЧКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

Нікіта _____

МАСЛОВ

2020

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

БД – база даних.

ОС – операційна система.

CSS – Cascade Style Sheets (каскадні таблиці стилів);

HTML – Hyper Text Markup Language (мова гіпертекстової розмітки).

API, ***Application Programming Interfac***e – набір використовуваних функцій, інтерфейс для створення програмних додатків.

RESTful API – API, що використовує HTTP запити GET, PUT, POST та DELETE для забезпечення доступу до інформації всередині системи.

Back end – частина комп'ютерної системи або програми, яка відповідає за зберігання та маніпулювання даними.

Front end – інтерфейс для взаємодії між користувачем і back end.

Десктопна програма – програма, призначена для використання на настільних комп'ютерах.

Chrome V8 engine – програма, що оброблює скрипти JavaScript у браузері.

Couroutines, корутини – структура управління виконання коду, схожа на потоки, що має свій власний стек, свої власні локальні змінні та дозволяє виконувати певною мірою паралельний код.

Callback, колбек – у Node.js, асинхронний еквівалент функції, що викликається по закінченні певної задачі.

DOM, Document Object Model – інтерфейс, що дозволяє програмам та скриптам динамічно отримувати доступ і оновлювати вміст, структуру та стиль html документа.

ВСТУП

У сучасному суспільстві волонтерські роботи грають важливу роль. Добровільна безкорислива допомога людям або тваринам, яким вона потрібна, йде на користь суспільству та допомагає зекономити гроші.

Тому не дивно що в США та в деяких країнах Європи є практика, яка впроваджує волонтерську діяльність учням та студентам. Вона полягає у тому, що для переходу на наступний рік навчання, учню потрібно займатися волонтерською діяльністю деякий час, наприклад 20 годин.

Але у цій системі виникають проблеми, які дуже заважають волонтерам. Наприклад складний документообіг: учню потрібно взяти документи у школі, знайти місце для проходження волонтерських робіт, де зможуть заповнити документи, та після проходження віднести документи до навчального закладу для перевірки, де директор має по документам затвердити проходження учнем волонтерських робіт. Але учень може підробити документи, та і сам пошук таких робіт може буди складною задачею для учня.

Отже, створення програмної платформи, яка вирішує проблеми сучасної системи, є актуальною задачею.

Даний дипломний проєкт присвячено розробленню програмної платформи для обліку волонтерської роботи, який має облегшити та покращити сам процес обліку волонтерських робіт учнів навчальних закладів.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

1.1. Аналіз процесу волонтерської роботи

Волонтерство, або волонтерська діяльність – безкорислива та добровільна діяльність, яка здійснюється окремими людьми або організаціями. Діяльність направлена на користь суспільству або окремим людям. Волонтерство є однією з форм благодійності і однією з підвалин громадянського суспільства.

На сьогоднішній день волонтерські роботи серед дітей та молоді не дуже популярні в силу того, що є більш цікаві речі для дозвілля та волонтер, у більшості випадках, не отримує нічого. У США знайшли спосіб покращити становище волонтерських робіт серед молоді. Для переходу у наступний клас або курс навчання учню потрібно набрати мінімально 20 годин волонтерських робіт. Таким чином хотіли долучити до волонтерських робіт учнів шкіл та студентів.

Але це створило інші проблеми. Опис основних проблем системи обов’язкових волонтерських робіт учнів та студентів можна побачити на дереві на рис. у вигляді ієрархічної структури.

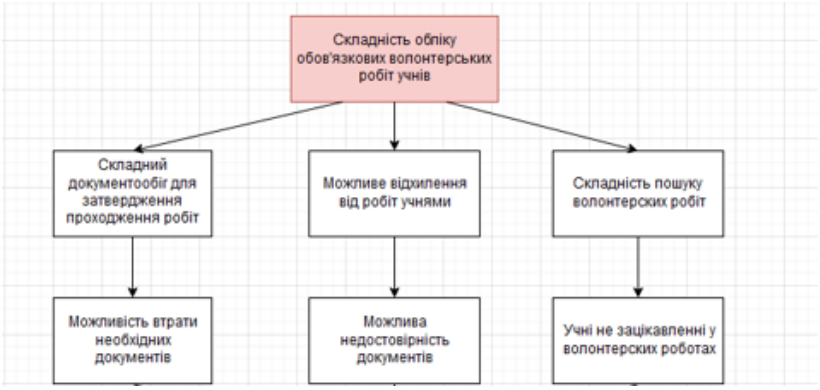


Рис. 1. Дерево проблем системи обов’язкових волонтерських робіт учнів та студентів

Правильній роботі системи заважають проблеми, які виникли, а саме:

1. Складний документообіг для проходження та затвердження проходження робіт – учню або студенту потрібно складати та заповнювати багато документів, які підтверджують те, що він був присутній на волонтерській роботі та успішно її виконав. Директору навчального закладу, в свою чергу, потрібно прийняти та перевірити усі документи, а також перевірити не фальшиві чи вони, а саме зв’язатися з відповідальним обличчям та з'ясувати за учня та його роботи.
2. Можливе ухилення від робіт учнями – з попередньої проблеми випливає інша, а саме те, що учні здатні підробляти документи та відхилятися від обов’язкових волонтерських робіт.
3. Складність пошуку волонтерських робіт – учні та студенті не завжди знають де саме шукати волонтерські роботи, де можуть підтвердити те, що він успішно виконав її. Також через складність пошуку, учні та студенті не зацікавлені у їх пошуку, тому вони беруться за перше що знайшли, а не за те, що може бути їм цікаво.

Умови для вирішення проблеми – наявність системи для контролю та пошуку волонтерських робіт з подальшим підтвердженням успішного їх виконання, яка вирішує проблеми які описані вище. Також система може збирати дані взаємодії волонтерів з волонтерськими роботами для подальшого аналізу, що може допомогти в розповсюдженні та популяризації волонтерських робіт серед молоді. Сама система та популяризація волонтерських робіт мають змогу зменшити кримінальні події серед молоді та дадуть змогу їм допомогти нужденним або суспільству.

1.2. Аналіз існуючих програмних додатків для волонтерів

1.2.1. Мобільний додаток “Goody. Помощь рядом”

Goody – геолокаційні співтовариство взаємодопомоги. Це сервіс, де кожен може допомогти іншому конкретною справою, об'єднатися і об'єднати ресурси для більшої допомоги. І, звичайно, отримати допомогу сам.

Цей додаток створений для того, щоб люди могли допомагати одне одному. Тобто, через Goody можна знайти людину якій потрібна допомога та допомогти, в свою чергу, користувач сам може залишити запит на отримання допомоги.

Переваги програмного додатка:

- можливість онлайн та по геоданих дізнаватися кому потрібна допомога;
- можливість самому отримати допомогу.

Недоліки програмного додатка:

- додаток не розповсюджений;
- додаток працює лише у деяких городах.

Цей програмний додаток не є вирішенням проблем , також через ряд недоліків використання даного додатка є проблематичним.

1.2.2. Мобільний додаток “Volunteer Abroad – GivingWay”

GivingWay – глобальна платформа, яка з'єднує волонтерів та організації по всьому світу – прямо та безкоштовно. Без посередників та оплати з’єднання.

Багато людей, які бажають поїхати волонтером за кордон, відмовились від цієї ідею через великі витрати часу та грошей.

За допомогою програми GivingWay Volunteer ви можете шукати організації, що охоплюють Південну Америку, Азію та Африку.

Переваги програмного додатка:

- безкоштовно;
- можливість самому обирати організації;
- можливість потрапити за кордон;
- можливість спілкування напряму з організацією.

Недоліки програмного додатка:

- додаток не розповсюджений;
- додаток ненадійний;
- можливість проходити волонтерські роботи лише за кордоном.

Цей програмний додаток не є вирішенням проблем , також через ряд недоліків використання даного додатка є проблематичним.

1.2.3. Мобільний додаток "Be My Eyes – помощь незрячим"

Незрячі і слабозорі люди можуть попросити допомоги у зрячого волонтера, який отримав повідомлення на телефон. Як тільки зрячий доброволець виконує дзвінок, між волонтером та незрячої людиною встановлюється аудіо- та відеозв'язок. Зрячий помічник може виявити допомогу по відеозв'язоку, граючи роль очей незрячої або слабозорої людини за допомогою камери свого телефону.

Незрячому або слабозорій людині може знадобитися абсолютно різна допомога, наприклад, дізнатися термін придатності молока або зрозуміти, чи поєднується одяг.

Переваги програмного додатка:

- реальна допомога незрячій або слабозорій людині;
- зручність використання.

Недоліки програмного додатка:

- додаток ненадійний;
- мало незрячих людей зможуть користуватися цим додатком.

Цей програмний додаток не є вирішенням проблем , також через ряд недоліків використання даного додатка є проблематичним.

Таблиця 1

Порівняльна характеристика програмних додатків

Назва додатка	Інтерфейс	Технології	Обробка даних	Функції подій та нагадування	Оцінка в App Store/ Play Market
Goody.	Створено без вимог	нативна розробка (Android)	відсутня	присутні	4/5
Volunteer Abroad – GivingWay	Створено без вимог	Крос-платформна розробка	відсутня	відсутні	3.6/5
Be My Eyes	Створено без вимог	нативна розробка (Android)	відсутня	присутні	3.9/5

1.3. Аналіз вимог до функціональності програмних засобів

Проаналізувавши сферу мобільних додатків та наявні аналоги програмного забезпечення для обліку волонтерської роботи, можна виділити наступні функціональні вимоги для розроблюваного дипломного проєкту:

1. Система має давати змогу директорам шкіл реєструвати свої школи.
2. Система має давати змогу директорам шкіл додавати та видаляти учнів своїх шкіл.
3. Система має давати змогу директорам шкіл створювати події для волонтерів (учнів та студентів).
4. Система повинна включати функціонал для редагування подій
5. Система має надавати волонтерам змогу переглядати події за допомогою мобільного додатка.
6. Система має надавати волонтерам змогу відмічати факт того, що вони були на події з додатковою інформацією – характеристики

події (назву, адресу, категорію), дату та час перебування волонтера на події.

7. Система має давати змогу директорам бачити відмітки волонтерів про події.
8. Система має давати змогу директорам підтверджувати або відхиляти відмітку.
9. Система повинна вести облік часу для волонтерів.
10. Система повинна надавати інформацію для волонтерів про їх прогрес у набутті необхідної кількості часу.
11. Система має показувати директору прогрес усіх його учнів.

Виділимо нефункціональні вимоги до розроблюваної системи:

1. Користування системою має бути доступно користувачу після того, як користувач погодився із правилами користування.
2. Інтерфейс користувача має бути простий для сприйняття.
3. Дизайн веб-сторінки має бути коректно показаний у основних веб-браузерах (Google Chrome, Mozilla Firefox, Safari, Microsoft Edge).
4. Мобільний додаток повинен коректно працювати на мобільних девайсах з ОС Android та iOS.
5. Збір даних про взаємодію волонтерів з подіями для подальшого аналізу.
6. Особисті дані користувачів повинні бути захищені.

1.4. Висновки до розділу

В результаті аналізу цієї предметної області можна зробити висновок, що існує необхідність створення мобільного додатка для контролю волонтерських робіт учнів та студентів.

Можна сформулювати задачі, які повинен вирішувати програмна платформа для обліку волонтерської роботи:

- впровадження системи створення та редагування подій для учнів та студентів – директор повинен мати змогу створювати та редагувати події для того, щоб учні та студенти брали участь у подіях за допомогою мобільного додатку;
- впровадження системи обліку часу для учнів та студентів – система повинна сама рахувати, оброблювати та зберігати;
- затвердженні або відхиленні години робіт для кожного учня та студента;
- впровадження системи затвердження годин – директор повинен мати усю необхідну інформацію для затвердження або відхилення відміток часу.

Були проаналізовані наявні рішення та технології, які використовуються для розробки мобільних додатків. Було виділено функціональні та нефункціональні вимоги для дипломної роботи.

2. АНАЛІЗ ТЕХНОЛОГІЙ РОЗРОБЛЕННЯ, МОВ ПРОГРАМУВАННЯ ТА СПОСОБІВ ПОБУДОВИ ПРОГРАМНИХ ЗАСТОСУВАНЬ

2.1. Огляд трендів розвитку Інтернету

Інтернет є найрозповсюдженішою системою у світі для зв’язку, та обміну інформацією. Більшість людей у світі користуються інтернетом та його веб-ресурсами.

У всьому світі налічується більше 1,24 мільярда веб-сайтів [1].

Рис. 2. Кількість веб-сайтів в 2018 році [1]

На грудень 2017 налічувалося 4 156 932 140 інтернет-користувачів. В Азії проживає майже половина всіх інтернет-користувачів [1].

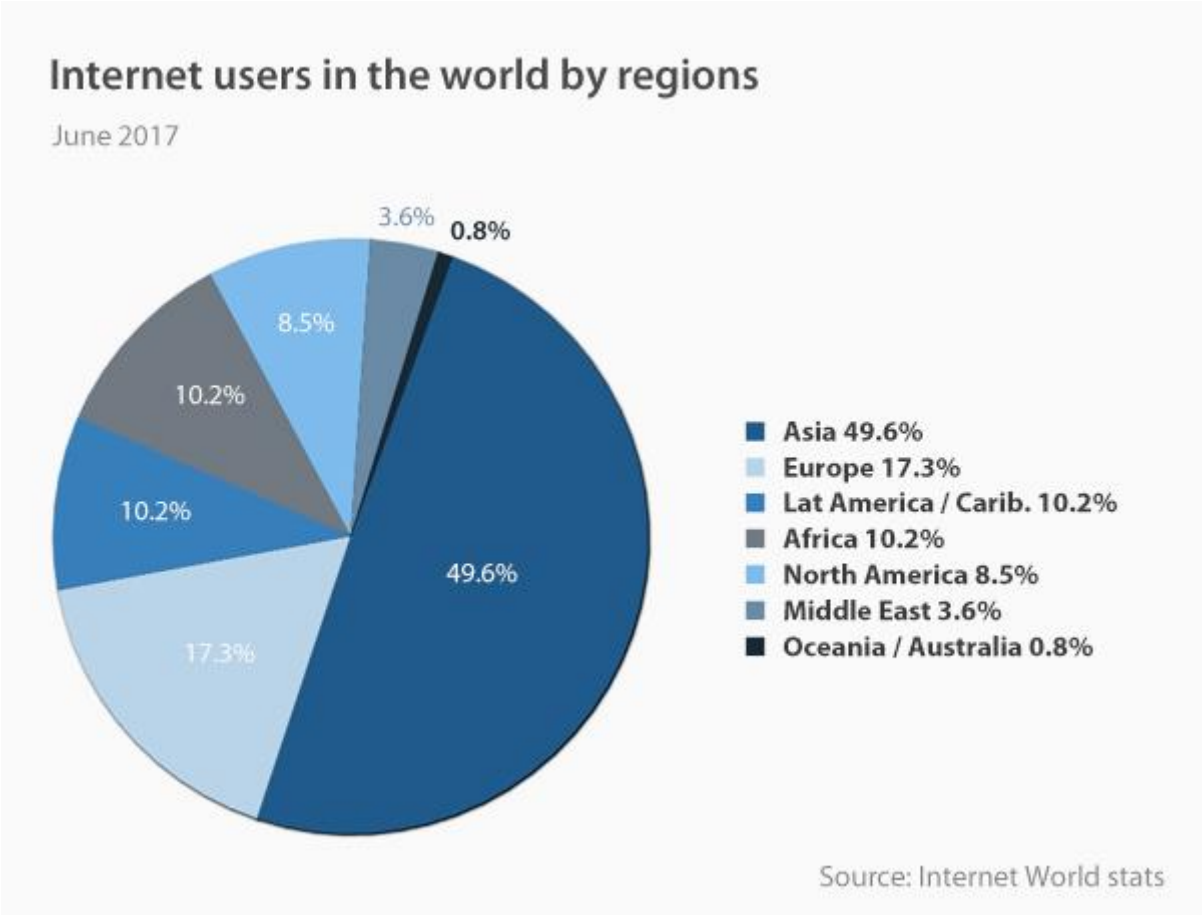


Рис. 3. Статистика по інтернет-трафіку [1]

Це показує наскільки веб-ресурси є розповсюдженими по всьому світі, що є не випадковим, бо якщо у користувача є підключення до інтернету, то набагато зручніше використовувати веб-сайт за допомогою браузера, а ніж встановлювати десктопну програму, у якої може бути відсутня крос-платформенность або інші проблеми у роботі додатку через системні та апаратні характеристики. В свою чергу, веб-сайти можуть не працювати коректно у різних браузерах, але зачасту розробники веб-застосунків оптимізують їх під всі браузери, що зробити набагато легше, ніж повністю оптимізувати десктопну програму.

Ще однією причиною популярності веб-сайтів є крос-платформеність інтернет браузерів. Це означає, що якщо на пристрої є інтернет браузер, то ви можете відкрити будь-який веб-сайт та користуватися їм. Але тут треба

зазначити, что не всі сайти оптимізовані під різну розподільну здатність екрану, що може сказатися у тому, як буде показаний веб-сайт, наприклад, на мобільному телефоні, або навіть сказатися на працездатності веб-сайту на тому самому мобільному телефоні.

У наш час дуже розвинені та розповсюджені бездротові інтернет технології: Wi-Fi та мобільний інтернет (2G, 3G, 4G та починається розповсюджуватися 5G). Ці технології, а також розвиток мобільних пристроїв за останні 20 років, призвело до того, що мобільні пристрої є у більшості людей у світі [1].

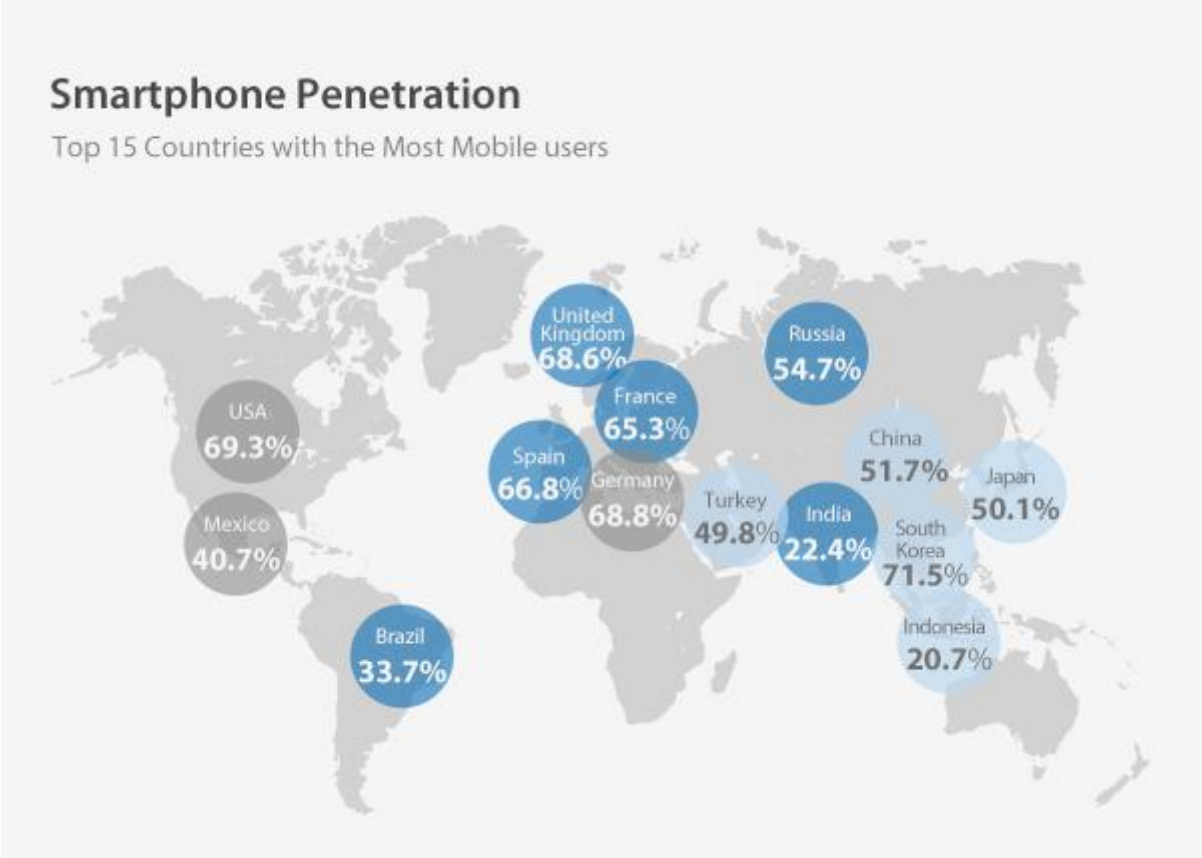


Рис. 4. Статистика по інтернет-трафіку [1]

Деякі прогнозували, що 2014 рік стане "революційним роком", коли кількість користувачів мобільних пристроїв перевершить кількість користувачів ПК; як виявилося, прогнози збулися – але в 2015 році і для США. З початку 2017 року мобільні пристрої стали популярнішими

настільних комп'ютерів, і на сьогоднішній день все залишається так само [1].

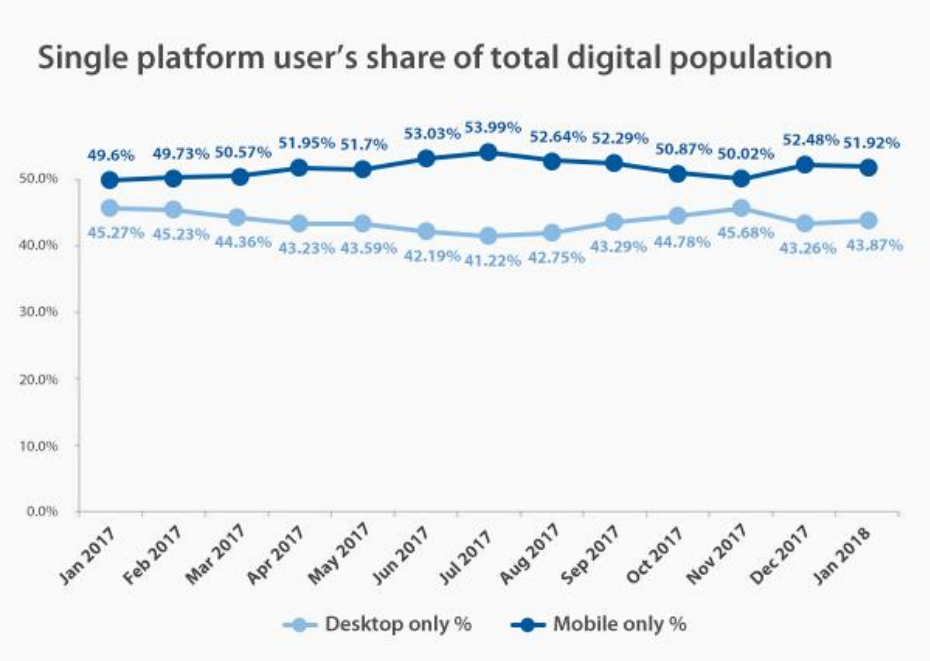


Рис. 5. Кількість користувачів смартфонів [1]

Це сприяє тому, що багато інтернету трафіку проходить через мобільні пристрої.

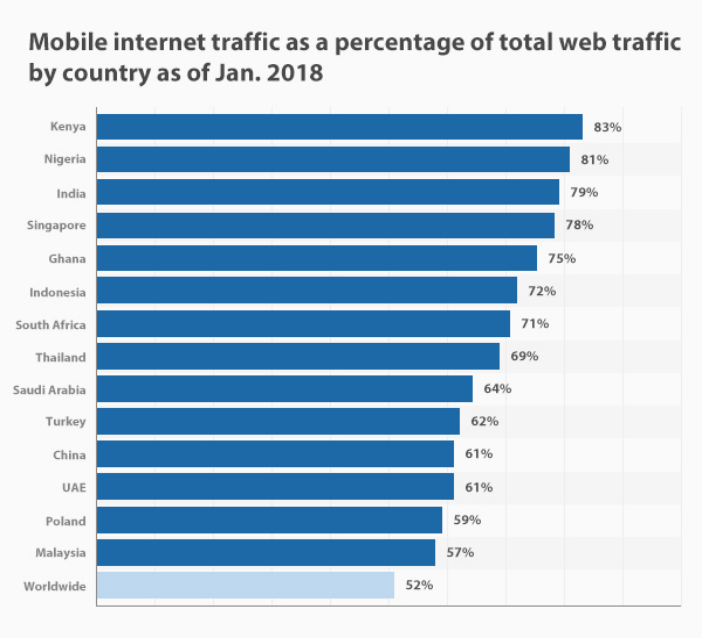


Рис. 6. Процентне співвідношення мобільного інтернет-трафіку [1]

Однією з головних проблем користування інтернетом була швидкість його підключення, але за останні роки вона тільки росте.

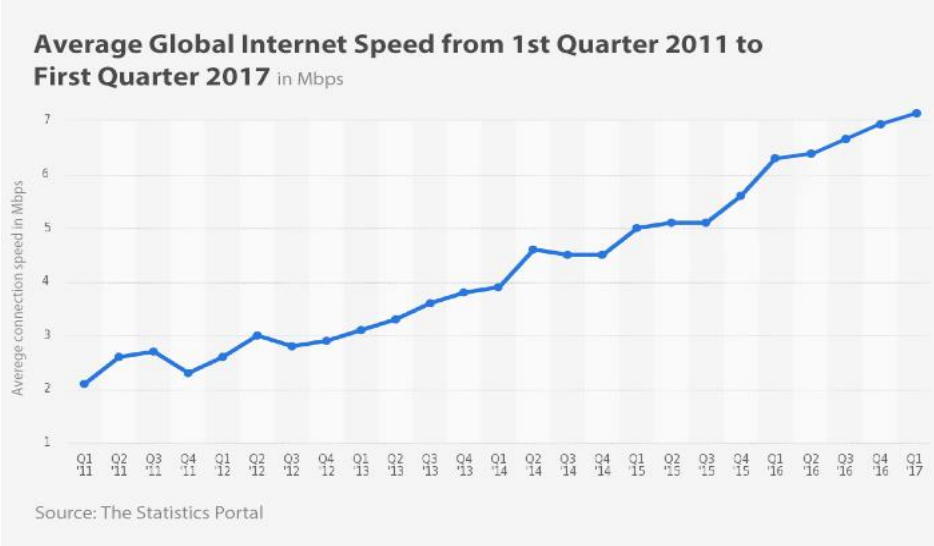


Рис. 7. Швидкість світового інтернету [1]

Це сприяє росту кількості користувачів інтернету на мобільних пристроях та кількості часу, які вони проводять в інтернеті в день.

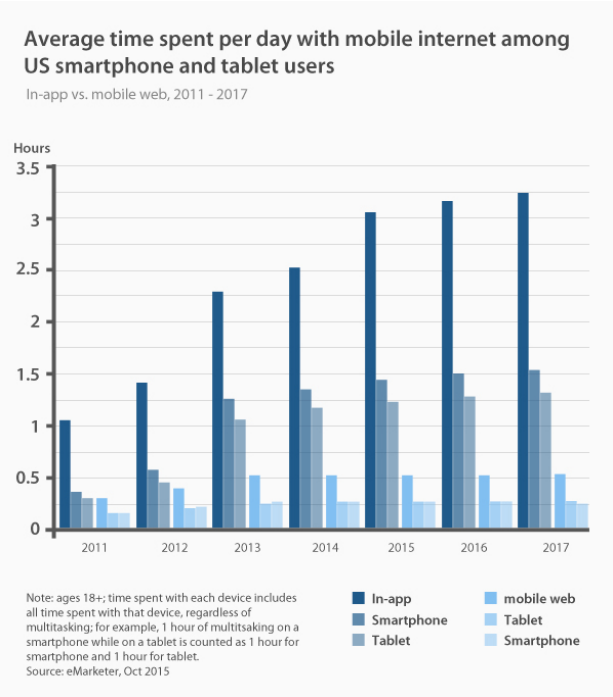


Рис. 8. Мобільні додатки vs. мобільний інтернет [1]

Кількість користувачів мобільних пристроїв в світі зростає і, згідно з прогнозами, до 2020 року 6,1 мільярда людей або 70% населення Землі будуть користувачами смартфонів [2].

2.2. Огляд можливих підходів до реалізації розроблюваної системи

Вибір способу коректної реалізації програми є важливим етапом у проєктуванні, бо програміст повинен знати які інструменти необхідні для розробки.

Розглянемо два можливі типи додатку для директорів шкіл: десктопна програма та веб-додаток. Клієнтська та серверна частини можуть застосовуватися у обох випадках.

Веб-застосування мають кілька переваг :

1. При написанні веб-застосування не треба значних зусиль та обчислювальних потужностей при розгортанні на машині клієнта. Для доступу до такого типу програм необхідний лише веб-браузер.
2. Такі додатки оновлюються частіше, однак програміст сам оновлює додаток, тому користувачам не потрібно локально його оновлювати. Також, виконуючи регулярні оновлення, програмісту, що розробляє веб-застосунок, легше знаходити та виправляти помилки в програмі.
3. Платформа є незалежною – розробник може не звертати на фреймворки та бібліотеки, що відрізняються у різних операційних засобах.
4. Є змога впровадження адаптивного дизайну на екранах із різним розширенням, у тому числі на мобільних пристроях.
5. Завдяки більш простому рівню освоєння, швидкість та вартість процесу розробки веб-додатку менша, ніж настільного додатку за рахунок нижчого рівня кваліфікації розробників.

Але веб-застосування мають також недоліки:

1. Залежність користувача від підключення до інтернету.
2. Низький рівень безпеки персональних даних користувача, оскільки всі дані зберігаються у віддаленій базі даних, що частіше за все знаходиться у незашифрованому хмарному сховищі.
3. Не всі браузери можуть підтримувати технології, які використані у веб-застосування.

А у свою чергу, програми для настільних комп'ютерів мають такі переваги:

1. Десктопні програми мають можливість зливатись з апаратною частиною, наприклад, із відеокартою чи аудіокартою, через системні оповіщення операційної системи.
2. Для користувача набір засобів операційної системи може бути значно зручнішим, ніж інтерфейс браузера.
3. Для веб-застосувань потрібно реєструвати сервер у мережі, а для десктопної програми потрібно лише використати інсталятор – що є значно зручним для кіннцевого користувача.
4. Використання десктопної програми значно дешевше для користувача. В свою чергу, для веб-застосунку власник має сплатити за сховище даних, хостинг та ін.
5. При поганій якості підключення користувача веб-застосування може працювати повільніше ніж, ніж програма.

Забезпечення для персональних комп'ютерів також має низку недоліків:

1. Десктопна програма має обмежений доступ, що обмежується одним кінцевим пристроєм.
2. Програма реалізує та користується для роботи своїм власним протокол, замість того, щоб користуватись готовим та універсальним протоколом роботи.

Додаток для волонтерів може бути розроблений у вигляді або сайту з адаптивним дизайном, або мобільної версії сайту, або мобільного додатку в силу того, що волонтери повинні будуть використовувати додаток там, де

використання комп’ютера буде неможливим або складним, а використання мобільного телефона буде зручним.

Для початку розглянемо версію сайту з адаптивним дизайном та мобільною версією сайту. Мобільний сайт – це спеціально створена версія сайту, яка розташована на спеціальному домені та яку бачать користувачі, коли заходять на вашу сторінку за допомогою браузера, встановленого на смартфоні або планшеті. На відміну від основної версії сайту, версія сайту для мобільних пристроїв адаптована під перегляд на невеликих екранах. Сайт з адаптивним дизайном має можливість змінювати дизайн та вміст сторінок веб-застосунку в залежності від розподільної здатності екрану та його розміру.

Сайт з адаптивним дизайном має кілька переваг :

1. Можливість показувати користувачам різний контент в залежності від пристрою.
2. Простота внесення змін. Якщо ви хочете додати новий блок на сайт, це потрібно буде зробити тільки один раз.
3. SEO-переваги. З адаптивним сайтом вам не потрібно буде займатися залученням трафіку на мобільну версію сайту.

Але також має свої недоліки:

1. Потрібно оптимізувати сайт для швидкого завантаження на мобільних пристроях.
2. З'являється необхідність готувати зображення різного розміру для настільних і мобільних пристроїв.
3. Для пристроїв з маленьким екраном буде потрібно створити окреме компактне меню.
4. Збільшує час тестування сайту.

В свою чергу, мобільна версія сайту має свої переваги:

1. Швидкість. Мобільний сайт швидко завантажується на смартфонах та планшетах.

2. Видимість в пошукових системах. Використовуючи окремий мобільний сайт, можна «заточувати» його під пошукові запити користувачів мобільних пристроїв.

А має і недоліки:

3. Кілька URL-адрес. Робота з декількома сайтами займає більше часу, а також вимагає додаткових витрат на SEO-оптимізацію.
4. Відсутність універсальності. На сенсорних і клавіатурних телефонах сайт буде виглядати по-різному.
5. Вартість створення мобільних сайтів значно вище, ніж вартість розробки мобільних додатків на замовлення.

Але варто розуміти, що додаток для волонтерів буде використовуватися лише на мобільних пристроях, тому робити додатковий сайт та оптимізувати його для роботи на мобільних пристроях не є цілком доцільно, бо це можна реалізувати за допомогою мобільного додатку.

Мобільний додаток має свої переваги:

1. Досвід взаємодії. Мобільний додаток має кращий UX-дизайн і досвід взаємодії з користувачами.
2. Доступ і швидкість. Додатки можуть працювати на мобільному пристрої навіть без підключення до інтернету. Таким чином, ваша інформація завжди буде доступна користувачеві. Швидкість завантаження також значно вище у додатки, ніж у сайту.
3. Видимість додатки на мобільному пристрої. Після його установки воно завжди знаходиться на екрані, а це підвищує шанси того, що користувач буде заходити частіше.

Недоліком є те, що додаток потрібно реалізувати та оптимізувати під 2 операційні системи: Android та iOS.

Оцінивши функціональні та нефункціональні вимоги до програмної розробки у підрозділі 1.3, а також визначивши переваги та недоліки деяких типів програмного забезпечення, описаних вище, було обрано реалізацію програмного застосунку для директорів у формі веб-додатку, а додаток для

волонтерів у формі мобільного додатку. Обидва додатка будуть з розділенням на клієнтську частину та серверну частину. Серверні частини для додатків будуть спільними.

2.3. Засоби програмування для back end частини ПЗ

Серверна частина веб-додатку складається із коду, що приймає та опрацьовує дані від front end частини, оброблює запити до бази даних, виконує більшу частину логічних операцій програми.

Найпоширенішими мовами програмування для розроблення веб-додатків на початок 2020 року є JavaScript, PHP, Java та Python [3]. Ці мови програмування мають безліч перевірених часом фреймворків, велику аудиторію підтримки та багато прикладів коду та робочих рішень.

PHP та Java можно використати як основну мову програмування для написання back end (серверної) частини веб-застосунку: використовуючи програмну платформу, Apache Struts, Hibernate, чи можливо Spring написати веб-додаток на мові Java, а із допомогою Symfony, Laravel чи Yii2 можна створити веб-застосунок на мові PHP. Але через брак досвіду розробки за допомогою цих мов, подальшу увагу зверну на JavaScript та Python.

Популярність мов програмування JavaScript та Python зростає з кожним роком серед розробників веб-додатків. На кінець 2018 року найуживанішими фреймворками для веб-розроблення додатків на мові Python були Flask та Django, а для розроблення на мові JavaScript – фреймворки, що були побудовані на основі Node.js: AdonisJs, Express.JS, Hapi.js тощо [4, 5].

Важливим пунктом для порівняння цих мов є швидкодія: наскільки користувач отримує відповідь на свої дії від застосунку. Порівнюючи швидкодію фреймворків Node.js із Python, можна помітити, що у Node.js швидкодія значно швидше. Це зумовлено тим, що Node.js базується на швидкому та потужному Chrome V8 engine. Крім того, фреймворки на мові Python працюють нестабільно в тих додатках, що вимагають великої кількості

оперативної пам’яті. Це говорить про те, що програмісту рекомендується використовувати Node.js у якості основного інструмента для розроблення back end частини застосунку, який працює з 3D-графікою.

Одним із аспектів для порівняння фреймворків Node.js із Python також є масштабованість та паралельна робота багатьох користувачів у веб-додатку. Масштабованість – це здатність програми обробляти зростаючу кількість запитів, не втрачаючи свою швидкодію. Ця властивість має серйозне значення у додатках з високою завантаженістю, що, наприклад, обслуговують велику кількість користувачів як із мобільних додатків, так із настільних комп’ютерів. Node.js при використанні створює однопоточну асинхронну архітектуру, у якій операції вводу та виводу завершуються за межами потоку, не блокуючи його. Ця властивість гарантує масштабованість Node.js у веб-додатках. Фреймворки, що написані на мові Python, не мають підтримки асинхронного програмування за замовчуванням, але все одно мають повний інструментарій для досягнення необхідної масштабованості веб-додатку, що досягається за допомогою корутин (coroutines).

Наступним пунктом для порівняння цих мов програмування в контексті розроблення веб-додатків є оброблення помилок. Простота та прозорість обробки помилок мають велику значимість у розробці. Як JavaScript, так і Python добре справляються із пійманням та обробленням помилок, що виникають під час виконання коду.

Важливим аспектом при виборі мови програмування є простота написання коду та швидкість освоєння нових технологій програмістами. На мою думку цей аспект є суб’єктивним, тому я віддаю перевагу JavaScript, через те, що маю досвід розробки на ньому та в мене не виникає проблем у освоєні нових технологій на JavaScript.

Веб-фреймворк Django, написаний на Python, легко масштабується, відмінно працює із різноманітними базами даних, має докладну та зрозумілу документацію та велику спільноту розробників. Але Django має

недолік – це його монолітність, тобто додаток неможливо розбити на декілька мікросервісів, Django не є зручним для реалізації потужних програм.

У разі використання Node.js може користуватися великою кількістю бібліотек від спільноти, показує більшу ефективність та швидкість, добре підходить для створення сервісів з API, легко обробляє одночасні запити та є популярним серед програмного забезпечення для серверної частини. Основним слабким місцем Node.js є його велика потреба у ресурсах процесора, оскільки Node.js працює в одному потоці. Ще Node.js має недолік який зв’язаний з самим програмістом, а сама програміст через недолік досвіду написання коду та використання колбеків (callbacks) може створити велику вкладеність, що має поганий вплив на розуміння коду та його швидкодії [6].

Таблиця 2

Порівняльна характеристика NodeJs та Django

	NodeJs	Django
Швидкодія	Базується на Chrome V8 engine	Нестабільно працює з великою кількістю оперативної пам’яті
Масштабованість	Створює однопоточну асинхронну архітектуру, у якій операції вводу та виводу завершуються за межами потоку, не блокуючи його	Немає підтримки асинхронного програмування за замовчуванням
Обробка помилок	Дозволяє обробляти програмні помилки та помилки програміста (дефекти написаного коду)	Дозволяє обробляти програмні помилки та помилки програміста (дефекти написаного коду)
Додаткові переваги	Простота написання коду (суб’єктивно). Велика кількість бібліотек від спільноти. Підходить для створення сервісів з API	Легко масштабується та відмінно працює із різноманітними базами даних
Недоліки	Велика потреба у ресурсах процесора, оскільки працює в одному потоці	Монолітність, тобто додаток неможливо розбити на декілька мікро-сервісів

2.4. Технології розроблення front end частини ПЗ

Front end розроблення, також відома як розроблення на стороні клієнта – це практика створення сторінок з розміткою HTML, стилістичних файлів на основі CSS та скриптів для веб-сайту або веб-застосунку на мові програмування JavaScript, для того, щоб користувач міг взаємодіяти із контентом веб-сайту [7]. Основною метою розробки front end частини додатку є забезпечення повної в обсязі та легкої для сприймання інформації при взаємодії користувача з контентом веб-сторінки у браузері, який розбитий на front end та back end частини .

Основними технологіями для створення клієнтської частини веб-сторінки для показу результатів роботи системи користувачу є HTML (Hypertext Markup Language), CSS (Cascading Style Sheets) та мова програмування JavaScript.

HTML – мова розмітки документів, що використовується для створення веб-сторінок для сайтів. За допомогою цієї мови розмітки розробник може показати вміст сторінки у вигляді блоків з параграфами, таблицями, формами для введення даних, картинками, аудіо та відео елементами тощо. HTML документ складається із серії коротких тегів, записаних автором сайту в текстовий файл. Веб-браузер зчитує файл та переводить текст у видиму форму, відтворюючи дизайн сторінки [8].

Визначення HTML складається із трьох наступних частин:

1. HyperText – гіперпосилання, за допомогою якоих користувач переходить веб-сторінками, натискаючи на них. Префікс «гіпер» у посиланні означає, що такий процес непрямий: є можливість переходити на різні частини Інтернету, перейшовши за різними посиланнями, без слідування певному порядку переходу сторінками.

2. Markup пояснює як HTML, за допомогою тегів показують інформацію. Теги визначають блокові структури сторінки, таблиці, медіа, тип тексту, наприклад, підкреслений чи курсивний тощо.
3. HTML є мовою розмітки зі своїм синтаксисом та власними визначеними словами.

Також важливим аспектом є CSS (Cascading Style Sheets) – засіб, за допомогою якого розробник може визначати привабливий дизайн створення та редагування елементів на веб-сторінці. Наприклад, стилі тексту, відступи навколо структурних блоків, прозорість картинки, розміри таблиць тощо. CSS дає розробникам більший контроль над виглядом веб-сторінки, аніж HTML. CSS допомагає уніфіковувати дизайн та використовувати його на декількох сторінках. Можливості CSS зростають із застосуванням таких компілюючих обробників шаблонів, як Sass чи LESS, що значно полегшує написання коду веб-розробниками дизайну [9].

JavaScript є потужним інструментом для розроблення front end частини веб-застосунку. JavaScript – це об’єктно-орієнтована скриптова мова, що використовується для надання веб-сторінкам динамічності та інтерактивності, наприклад, створення анімацій, модульних вікон, та також для написання логіки на стороні клієнта. JavaScript код на стороні клієнта дозволяє динамічно розміщувати та змінювати HTML елементи, отримувати та обробляти дані форм, реагувати на такі дії користувача, як введення інформації чи навігація по сторінках. JavaScript має змогу налагодити двосторонній зв’язок із серверною(back end) частиною додатку, за допомогою бібліотеки React та технології Ajax. За допомогою цих двох технологій можна динамічно оновлювати вміст веб-сторінки без фактичного оновлення сторінки користувачем, створювати запити до сервера, отримувати дані від сервера.

React це один з найпопулярніших фреймворків на мові програмування JavaScript, який дозволяє працювати з DOM-деревом. React дозволяє при зміні даних оновлювати сторінку без її перезавантаження, робити запити до сервера, працювати з даними на стороні клієнта [10].

Bootstrap – популярний фреймворк для створення front end частини веб-додатку. Цей фреймворк розроблений у Twitter та оснований на LESS CSS і jQuery. За допомогою Bootstrap можна реалізувати адаптивність веб-сторінок для можливості користування при різних розширеннях. Bootstrap правильно працює із найпопулярнішими браузерами: Google Chrome, Opera, Mozilla Firefox, Microsoft Edge, Safari.

Мобільний додаток також можна розбити на front end та back end частини. Back end частина, тобто сервер, можна використати той же, що використовується для веб-додатку. Для front end частини використовуються інші, від технологій для веб-додатку, технології: React Native, Mobx та JSX.

React Native – крос-платформенний фреймворк на основі React, що застосовується для написання мобільних додатків для пристроїв на базі Android та iOS. React Native має усі переваги React, велику кількість різноманітних розширень та велике ком'юніті. Різниця з React в тому, що React Native не маніпулює з DOM-деревом, а інтерпретує написаний розробником JavaScript код у код, який буде зрозумілий для нативної платформи(SDK) [11].

Mobx – бібліотека для Reac та React Native, головна функція якої створення та робота зі спеціальним сховищем інформації(state). Особливість полягає в тому, що можно точково оновлювати інформацію в сховище та точково змінювати вигляд та контент у додатку та не рендерити все спочатку. Це, у свою чергу, зменшує навантаження на додаток та покращує його швидкодію.

JSX – це подібне до мови шаблонів розширення синтаксису для JavaScript, але з усіма можливостями самого JavaScript. JSX складається з особливих шаблонів, які є більш зручними та гнучкими, порівняно з тегами HTML. Також розробник має змогу створювати та програмувати нові унікальні шаблони, які він може використати в будь-якому місті програми. При рендері інтерпритатор перетворить JSX на елементи розмітки HTML з скриптами на JavaScript [12].

3. ОПИС РОЗРОБЛЕНИХ ПРОГРАМНИХ ЗАСОБІВ

3.1. Загальна структура системи

Програмне забезпечення реалізоване у вигляді веб-застосування для директорів та мобільного додатку для волонтерів. Це забезпечення можна розділити на три частини: клієнтські частини веб-сайту для директорів та мобільного додатку для волонтерів (front end) та сервер (back end). Кожна частина має свої функціональні та нефункціональні вимоги, свою роль та виконує свої функції.

Серверна частина розробленого програмного забезпечення реалізована на основі патерну MVC та призначена для:

1. Оброблення запитів від клієнтської частини веб-застосування та від мобільного додатку.
2. Роботи з базою даних для збереження, зчитування, видалення та оновлення даних.
3. Проведення аналізу інформації взаємодії волонтерів з волонтерськими подіями.
4. Робота посередником для волонтерських подій між веб-додатком для директорів та мобільного додатку для волонтерів.
5. Розрахунку необхідної кількості годин для волонтера, яка визначена школою.
6. Систематизації даних по школі для конкретного директора.

У свою чергу клієнтська частина розробленої системи призначена для:

1. Надання зручного та зрозумілого інтерфейсу для взаємодії користувача із системою.
2. Створення асинхронних запитів до сервера для оновлення даних у веб-застосуванні чи мобільному додатку.
3. Візуалізації даних про школи, волонтерів та події у зручному для користувача вигляді.
4. Підтримки адаптивності для різних пристроїв з різними характеристиками екрану.

Архітектура розроблювальної системи складається із багатьох модулів, які пов'язані між собою та мають свої унікальні ролі у системі.

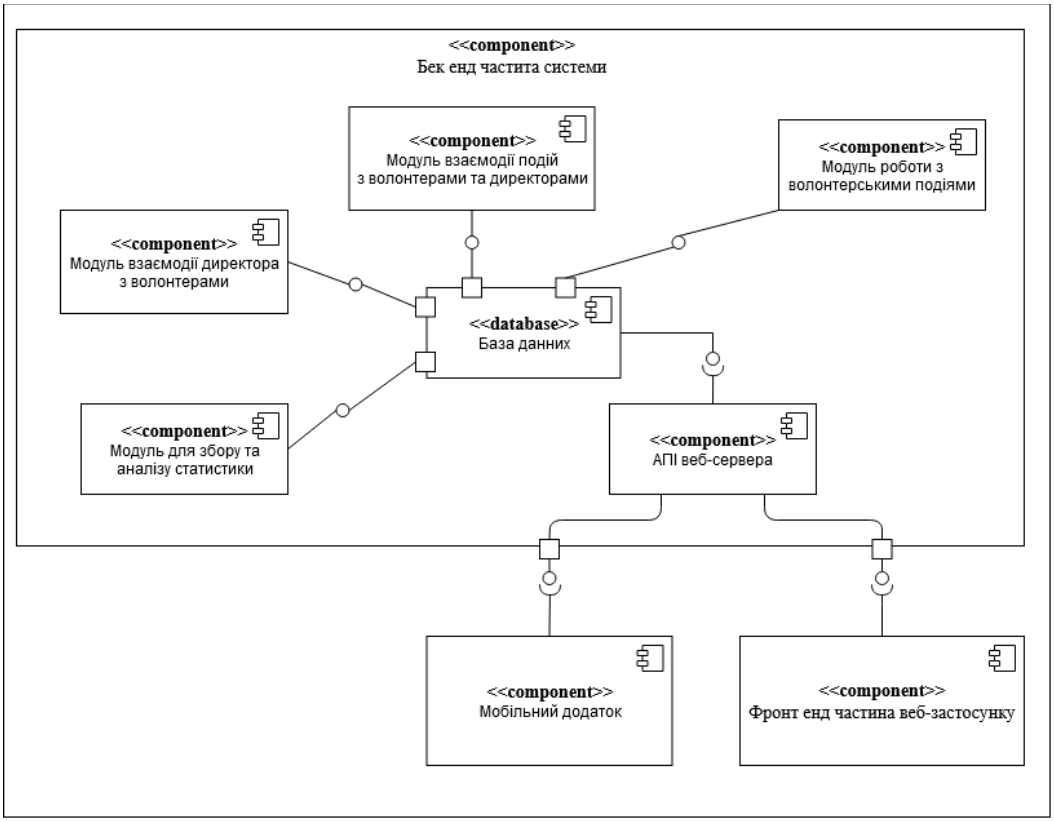


Рис. 9. Архітектура системи

У розробленій системі мають місце такі модулі:

1. Модуль роботи з волонтерськими подіями. Директор має змогу створювати події, видаляти їх та оновлювати інформацію про них.
2. Модуль взаємодії подій з волонтерами та директорами. Модуль забезпечує передачу створених директором подій до мобільного додатку волонтера та взаємодію волонтера з подією.
3. Модуль взаємодії директора з волонтерами. Цей модуль надає змогу директору імпортувати студентів до системи, видаляти їх та переглядати інформацію про проходження подій волонтерами.
4. Модуль API. Цей модуль забезпечує зв’язок між клієнтською та серверною частинами.

5. Модуль взаємодії із базою даних. Даний модуль надає змогу серверу проводити маніпуляції з даними у базі даних: видаляти, створювати, оновлювати записи.
6. Серверна частина. Цей модуль є центральним та основним у системі, тобто він пов’язує усі модулі один з одним та забезпечує взаємодію між ними.
7. Клієнтська частина. Цей модуль відповідає за взаємодію користувача з системою та відображення потрібної інформації.
8. Модуль для збору та аналізу статистики. Цей модуль відповідає за збір статистики по подіям та аналізу інформації.

Для серверної частини основною мовою програмування було обрано JavaScript, оскільки і клієнтська частина буде написана на JavaScript. Ще важливою причиною є те, що у розробника системи є достатній досвід у розробці програмного забезпечення на JavaScript.

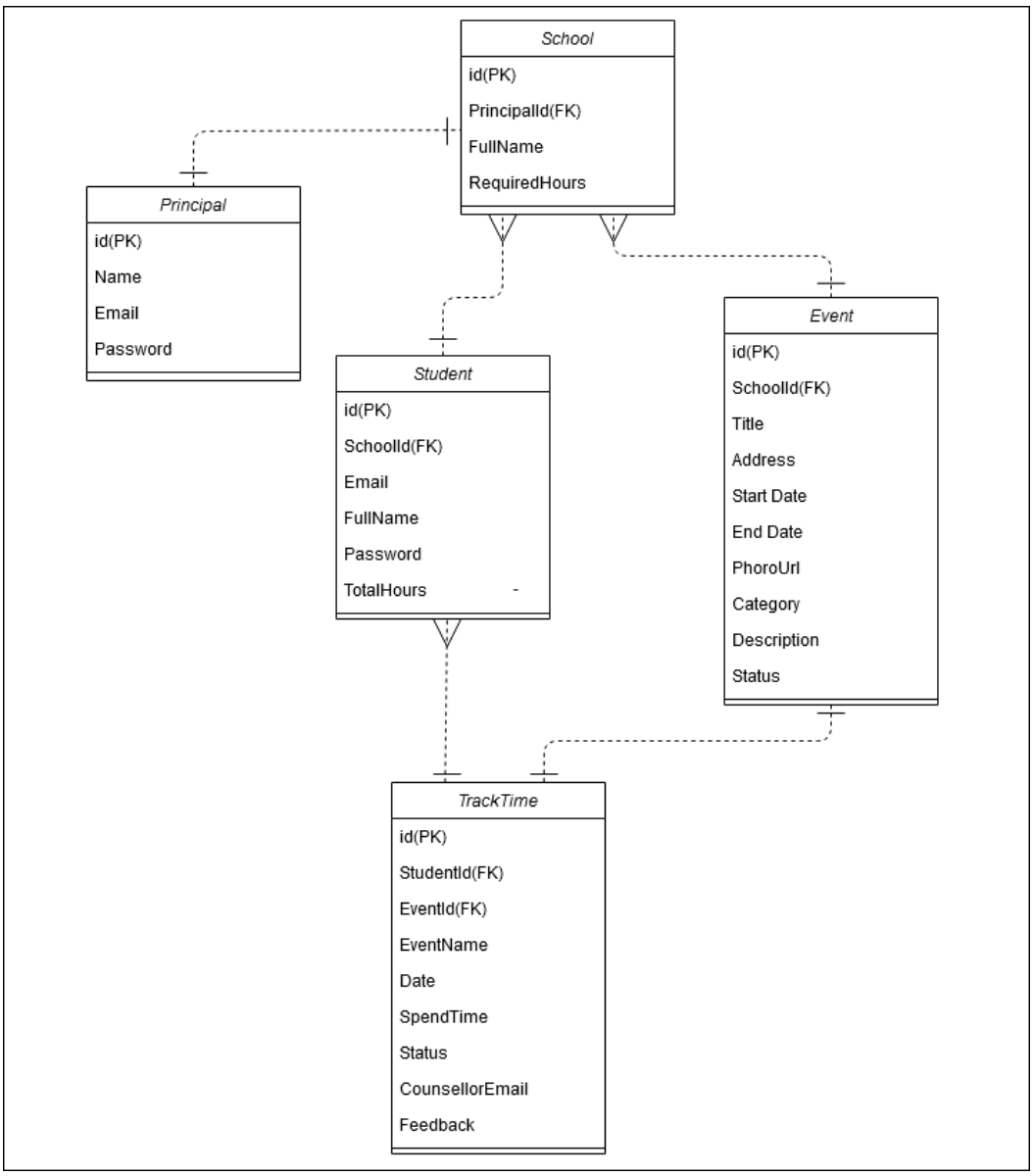


Рис. 10. Структурна бази даних

У базі даних визначені такі таблиці: Schools, Principals, Students, Events, TrackTimes. Усі таблиці нормалізовані у першу, другу та третю нормальну форму, тобто створено окремі таблиці для кожного набору даних, де дані визначені первинними ключами, було усунено однакові дані в таблицях, створені окремі таблиці для наборів даних, що відносяться до декількох записів, а ці таблиці у свою чергу пов’язані один з одним за допомогою зовнішніх ключів, було видалено поля, що не залежать від жодного ключа.

Таблиця Principals зберігає інформацію про директора:

1. Повне ім’я директора.

2. Логін директора(Email).
3. Пароль директора.

Таблиця Schools зберігає загальну інформацію про школу:

1. Посилання на директора в таблиці Principals.
2. Повна назва навчального закладу.
3. Потрібна кількість годин волонтерства для студента.

Таблиця Students зберігає інформацію про студента(волонтера):

1. Логін студента(Email).
2. Пароль студента.
3. Повне ім’я студента.
4. Середній бал.
5. Кількість годин, які студент витратив на волонтерські події.
6. Посилання на школу в таблиці Schools.

Таблиця Events зберігає інформацію про волонтерські події:

1. Назва події.
2. Адреса події.
3. Час та дата початку події.
4. Час та дата кінця події.
5. Фото події.
6. Категорія події.
7. Опис події.
8. Статус події.
9. Середній бал.
10. Пароль для відповідального
11. Посилання на школу в таблиці Schools.

Таблиця TrackTimes зберігає всі відмітки волонтерів про участь у волонтерській події:

1. Назву події.
2. Посилання на подію таблиці Events.

3. Посилання на волонтера в таблиці Students.
4. Дату відмітки.
5. Кількість часу перебування на події.
6. Статус відмітки.
7. Email відповідального.
8. Відгук студента.

3.2. Алгоритм роботи з подіями

Основними сутностями, з якими працює система, є волонтерські події. Директори шкіл мають змогу створювати волонтерські події, які розповсюджуються на всю школу. При створенні події, директор повинен вказати назву, адресу, де буде проводитись подія, дату проведення, час початку та тривалість події. Також директор має змогу оновлювати інформацію про подію. Але ця можливість доступна до того часу, який вказаний як початок події, тобто поки подія не почалася.

Після створення або оновлення події, вона становиться доступною для кожного зареєстрованого учня, тобто волонтера, у школі за якою закріплений директор який створив подію. Усі доступні події для волонтера будуть показуватися у мобільному додатку. Волонтер має змогу переглядати інформацію про події.

Після початку події, якщо волонтер прийшов на неї, він має змогу відмітити про прихід на подію, що призведе до того, що система почне рахувати кількість часу участі волонтера у події.

Перед тим як залишити подію, волонтер повинен зробити нову відмітку, щоб система зберегла кількість часу, проведеного волонтером на події. При цьому волонтер повинен заповнити додаткову інформацію: відгук про подію, оцінку самої події та email відповідального за подію.
Цей email потрібен для того, щоб директор зміг перевірити факт того, що волонтер приймав участь у події.

Також є додатково 2 поля для відповідального: пароль та оцінка для самого волонтера. При створенні події, директор має змогу задати пароль для відповідального, який він може використати щоб поставити оцінку волонтеру. Якщо відповідальних захоче когось виділити оцінкою, то він може просто ввести пароль, коли волонтер буде робити відмітку, та поставити оцінку. Без пароля оцінка не буде врахована. Ця система була впроваджена для того, щоб волонтер не міг змогу сам ставити собі оцінку.

Якщо подія закінчилася до того, як волонтер зробив відмітку про закінчення, система автоматично зробить відмітку, але без відгука про подію та email відповідального за подію.

Директор має змогу бачити усі відмітки учнів школи, за якою він закріплений. Нові відмітки будуть зі статусом “Awaiting”, тобто директор ще не переглянув їх. Директор змінити статус відмітки, тобто написати відповідальному за подію, щоб перевірити правильність даних про волонтерську роботу конкретного учня. Після цього директор може встановити новий статус для відмітки: або “Confirmed” – тобто відмітка підтверджена та час, якій був вказаний у відмітці, зарахується до загального часу учня, який він витратив на волонтерські події, або “Rejected” – тобто відмітка відхилена і не буде враховуватися.

Вся статистика по подіям та відміткам конкретного волонтера буде доступна у нього у мобільному додатку. Так волонтер може дізнатися про його загальну кількість годин, його відмітки та їх статус.

4. АНАЛІЗ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1. Особливості реалізації

Back end частина мобільного додатку та веб-додатку реалізована за допомогою архітектурного патерну MVC (Model View Controller). Дана архітектура складається з 3 частин: Модель(Model), Представлення(View) та Контролер(Controller). Модель відповідає за роботу з даними – валідація даних, їх обробка та зв’язок з базою даних. Представлення відповідає за получення необхідних даних з моделі та відправки їх користувачу. Контролер відповідає за зв’язок користувача з системою, направляє дані від користувача до системи і навпаки. Контролер використовує модель та представлення для реалізації необхідного функціоналу.

Модель і контролер знаходяться у серверній частині, а представлення – у клієнтській частині. Представлення спілкується з сервером за допомогою Контролера, який виконує певні операції та отримує потрібні дані від Моделі.

4.2. Дизайн та вміст сторінок додатків

Вміст та дизайн сторінок веб-додатку були реалізовані за допомогою технологій JavaScript, CSS та HTML5. Сторінки веб-додатку мають адаптивний дизайн, тобто мають змогу підлаштовуватися під розмір та роздільну здатність екрану користувача, що було досягнуто завдяки бібліотеки Bootstrap 5.

Вміст та дизайн сторінок мобільного додатку були реалізовані за допомогою фреймворку React Native. Цей фреймворк дає змогу за допомогою JavaScript створити компоненти та сторінки з дизайном, з яких буде складатися мобільний додаток.

Так, як веб-додаток та мобільний додаток використовують JSX, то сторінки кожного додатку розбиті на унікальні компоненти зі своїм дизайном та логікою, які можна використовувати у різних місцях цільового додатку.

У веб-додатку існують такі сторінки:

1. Головна сторінка з формою логіну та реєстрації.
2. Сторінка зі створеними подіями.
3. Сторінка для створення подій.
4. Сторінка з учнями в школі та їх відмітками.
5. Сторінка зі статистикою.

На кожній сторінці веб-застосунку присутній універсальний елемент, який виконує роль навігаційної панелі. Використовуючи цю панель користувач має змогу пересуватися по сторінкам за допомогою кнопок з гіперпосиланнями.

Головна сторінка містить форми для логіну та реєстрації директорів в системі. Форми змінюються за допомогою спеціальної кнопки без оновлення сторінки.

Форма логіну, яка зображена на рис. 11, пропоную користувачу ввести пару логін(електронна пошта) та пароль, щоб увійти у систему. Якщо логін або пароль будуть введені неправильно, то буде показано повідомлення про помилку.

Рис. 11. Головна сторінка з формою логіна

Форма реєстрації нового директора, яка зображена на рис. 12, містить поля для вводу електронної пошти, паролю та поле для підтвердження паролю. Після успішного проходження цього кроку, користувачу потрібно заповнити

поля з інформацією по школі – повна назва навчального закладу та потрібну кількість годин для волонтера.

Рис. 12. Головна сторінка з формою реєстрації

Сторінка зі створеними подіями, яка зображена на рис. 13, містить список всіх створених директором подій. По кліку на подію, користувач отримує повну інформацію про подію(назву, адресу, опис, фото тощо).

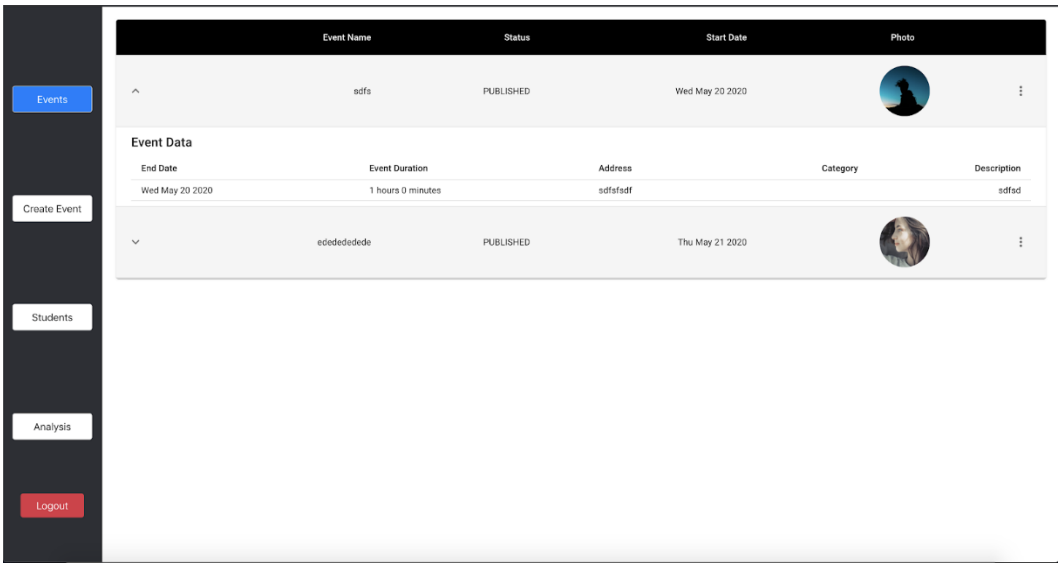


Рис. 13. Сторінка зі створеними подіями

Сторінка для створення подій, яка зображена на рис. 14, складається з форми з полями, яка дозволяє користувачу створити подію з потрібними полями. Користувач потрібен ввести ім’я події, її адресу, опис, вибрати фото,

вказати дату, час та тривалість події. Якщо всі поля заповнені правильно, то буде створено подію, яка буде відображена у мобільному додатку та на сторінки зі списком всіх подій веб-додатку.

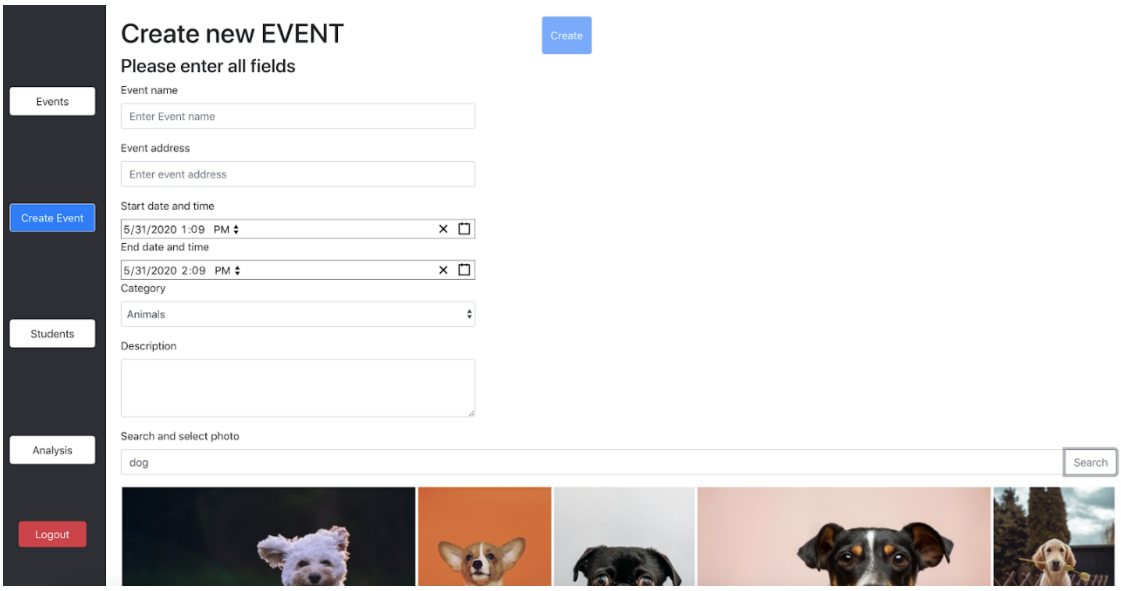


Рис. 14. Сторінка для створення подій

Сторінка з учнями в школі, яка зображена на рис. 15, складається зі списку всіх учнів, які відносяться до цієї школи. По кліку на учня, користувач отримує інформацію про учня та список його відміток. Кожна відмітка має 3 статуса: Pending, Rejected, Approve. Директор може змінювати статус відмітки.

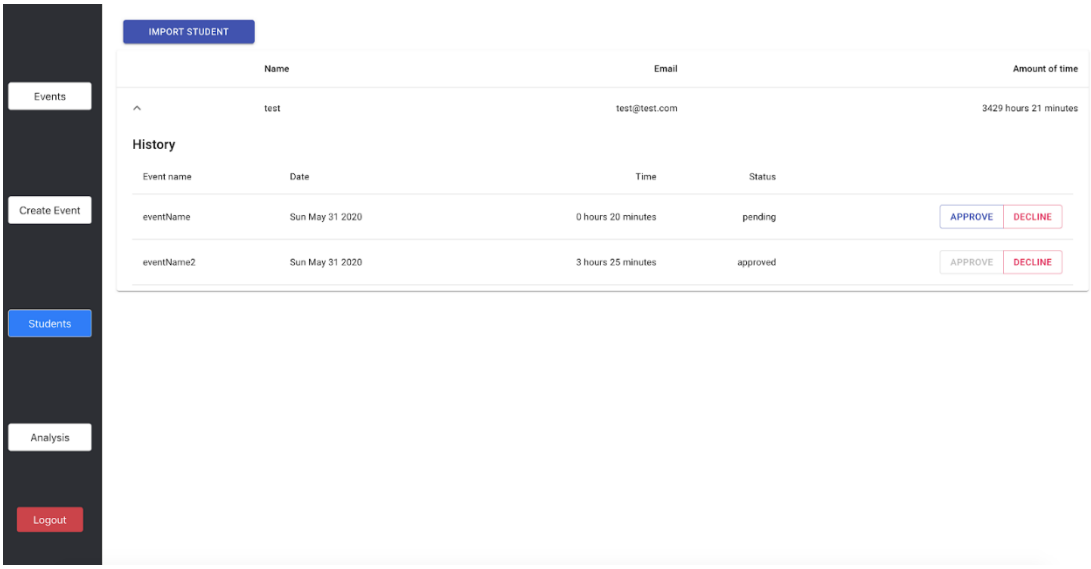


Рис. 15. Сторінка з учнями в школі

Сторінка зі статистикою показую користувачу, тобто директору, усю статистику по школі, а саме інформацію по усім волонтерам, подіям та категоріям.

У мобільному додатку існують такі сторінки:

1. Головна сторінка з формою логіна.
2. Сторінка реєстрації.
3. Сторінка з доступними подіями.
4. Сторінка з інформацією про подію.
5. Сторінка створення відмітки.
6. Сторінка профілю волонтера.

Головна сторінка містить форми для логіну та реєстрації волонтера в системі. Форми змінюються за допомогою спеціальної кнопки без оновлення сторінки.

Форма логіну, яка зображена на рис.16, пропоную користувачу ввести пару логін(електронна пошта) та пароль, щоб увійти у систему. Якщо логін або пароль будуть введені неправильно, то буде показано повідомлення про помилку.

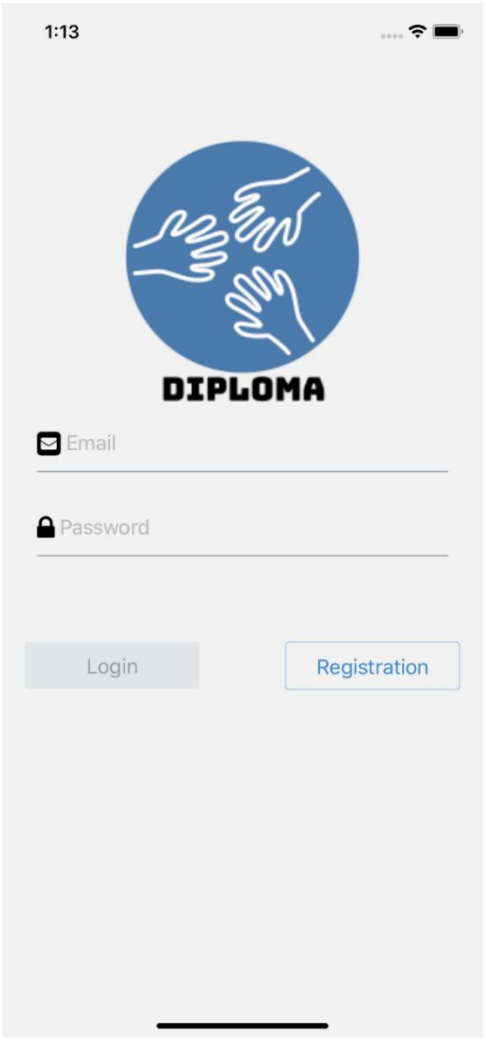
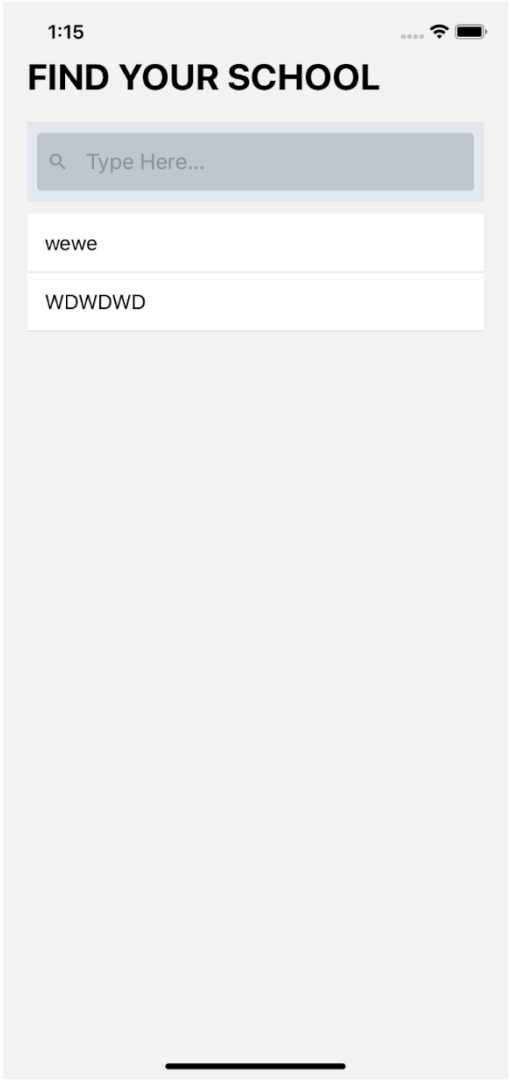


Рис. 16. Сторінка логіну

Сторінка реєстрації нового волонтера (рис. 17, 18) починається з того, що користувач має знайти та вибрати свою школу. Після цього учень потрапляє на сторінку реєстрації, де користувачу потрібно ввести електронну пошту, ім’я та пароль. Якщо директор не створив даного учня, то користувач отримує помилку.



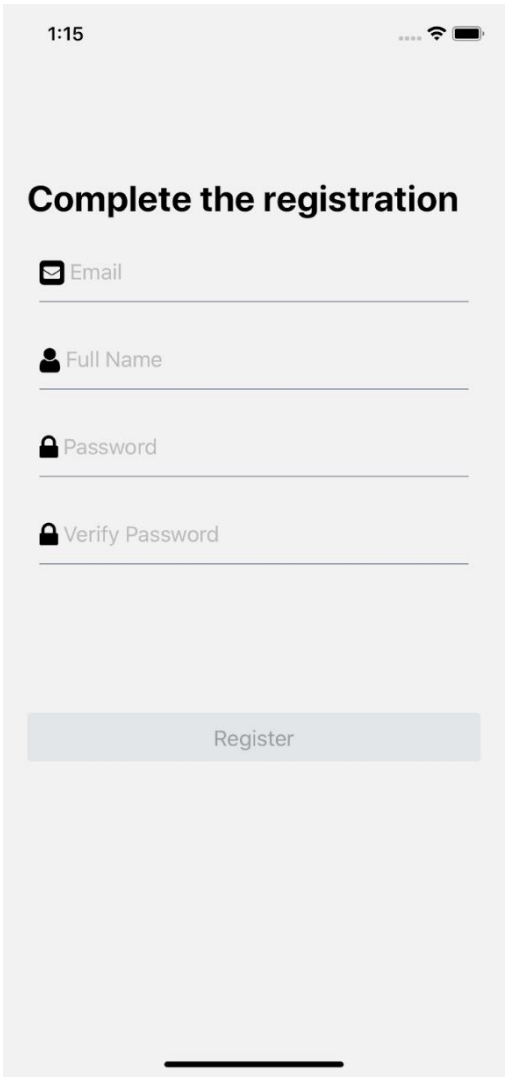


Рис. 17. Сторінка вибору школи Рис. 18. Сторінка реєстрації

Сторінка з доступними подіями, яка зображена на рис.19, складається зі списку доступних волонтеру подій. По кліку на подію користувач переходить до сторінки з інформацією про подію.

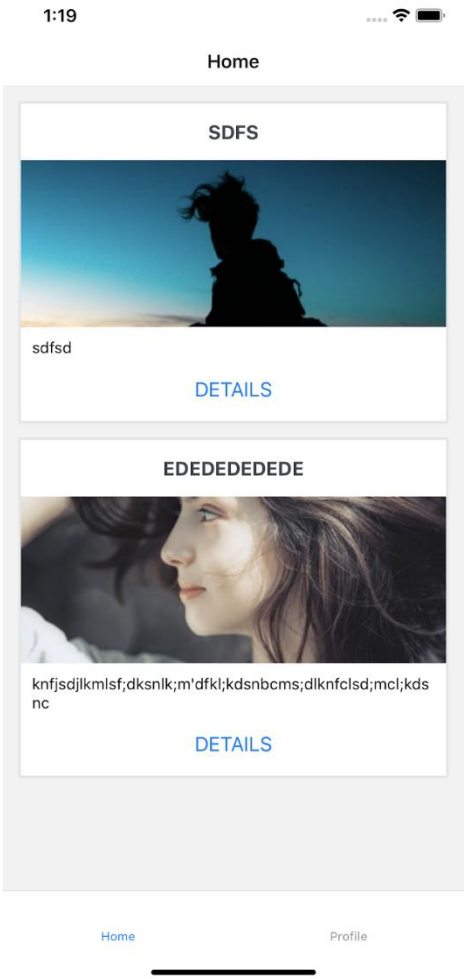


Рис. 19. Сторінка з доступними подіями

Сторінка з інформацією про подію, яка зображена на рис.20, складається з інформації про подію та таймеру часу, який волонтер бере участь у події. Таймер має кнопку Start, яку неможливо натиснути до початку події. Після натискання, таймер почне рахувати час, а кнопка зміниться на Stop, після натискання якої таймер збереже час, та користувач перейде до сторінки створення відмітки.

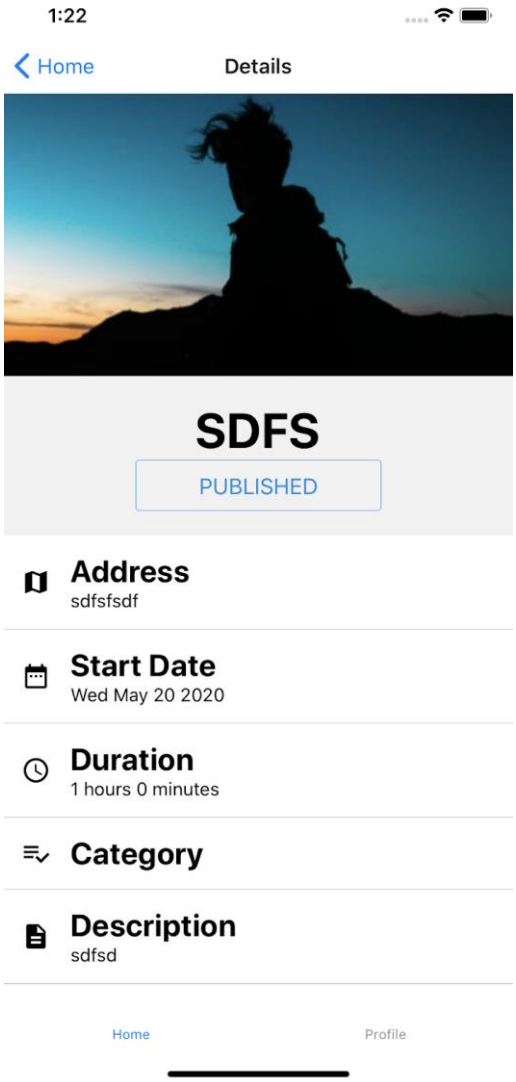


Рис. 20. Сторінка з інформацією про подію

Сторінка створення відмітки, яка зображена на рис.21, складається з кількістю часу, яку рахував таймер, поле для вводу відгуку про подію, поле для вибору оцінки події, поле для вводу пароля відповідального, поле для оцінки волонтера та поле для написання електронної пошти відповідального за подію, щоб після створення відмітки, директор мав змогу перевірити правдивість відмітки.

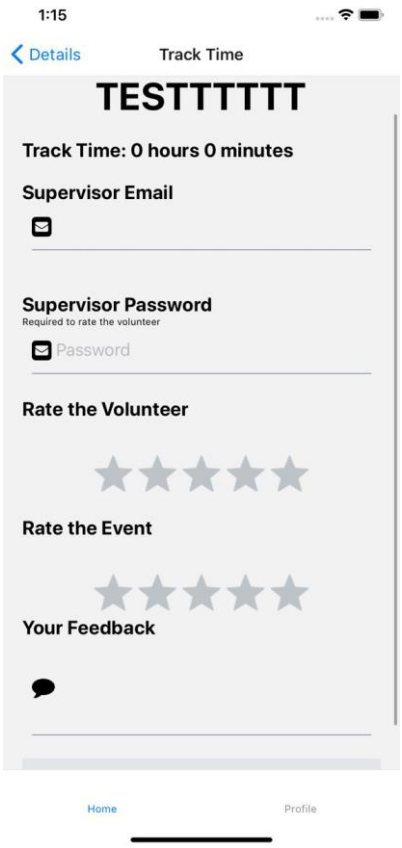


Рис. 21. Сторінка створення відмітки

Сторінка профілю волонтера, яка зображена на рис.22, складається з інформації про користувача та його відміток, де волонтер має змогу слідкувати за статусом його відміток.

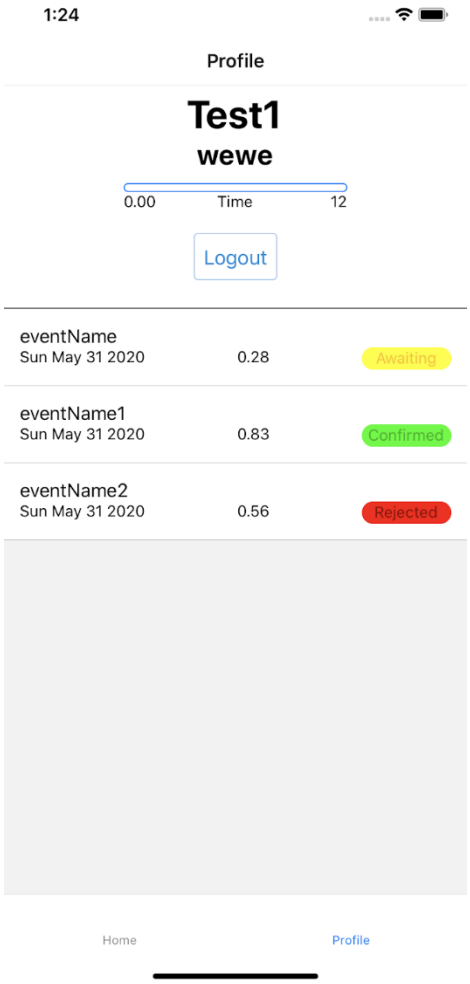


Рис. 22. Сторінка профілю волонтера

ВИСНОВКИ

Метою даного дипломного проєкту було розроблення програмної платформи для обліку волонтерської роботи.

Аналіз засобів розроблення веб-додатків та мобільних додатків, попередньо виконаний в дипломному проєкті, показав доцільність створення веб-додатку на платформі Node.js з використанням технології AJAX та створення мобільного додатку за допомогою React Native.

Розроблена програмна платформа:

– забезпечує авторизований доступ веб-додатку та мобільного додатку,

– дозволяє створювати, редагувати та видаляти волонтерські події,

– дозволяє волонтерам створювати відмітки про участь у волонтерських роботах,

– дозволяє директорам отримувати статистику по школі,

– підтримує динамічне оновлення вмісту web-сторінок,

– має зрозумілий та зручний інтерфейс.

Розробка виконана у повному обсязі, всі вимоги замовника враховані, тестування продукту виконано у відповідності до затвердженої програми та методики тестування.

Використання розробленої програмної платформи забезпечить директорів та волонтерів автоматизованим інструментом, що допоможе облегчити процес обліку волонтерської роботи.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРНИХ ДЖЕРЕЛ

1. Интернет–тренды 2020 года [Електронний ресурс]. — Режим доступу : https://ru.vpnmentor.com/blog/интернет–тренды–статистика–и–факты–в–с/?tldr=46717 Дата доступу: травень 2020.
2. 6.1B Smartphone Users Globally By 2020 [Електронний ресурс]. — Режим доступу : https://techcrunch.com/2015/06/02/6–1b–smartphone–users–globally–by–2020–overtaking–basic–fixed–phone–subscriptions/#.qmtrknq:RPIH Дата доступу: травень 2020.
3. The State of Developer Ecosystem in 2018 [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/research/devecosystem–2018. Дата доступу: травень 2020.
4. JavaScript in 2018 survey [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/research/devecosystem–2018/javascript. Дата доступу: травень 2020.
5. Python in 2018 survey [Електронний ресурс]. – Режим доступу: https://www.jetbrains.com/research/python–developers–survey–2018. Дата доступу: травень 2020.
6. Node.js Documentation 2019 [Електронний ресурс] – Режим доступу: https://nodejs.org/uk/docs/ Дата доступу: травень 2020.
7. Lindley Cody. Front–end Developer Handbook 2019 [Електронний ресурс] – Режим доступу: http://bit.ly/2WqhC9H. Дата доступу: травень 2020.
8. React [Електронний ресурс] – Режим доступу: https://ru.reactjs.org/docs/getting–started.html Дата доступу: травень 2020.
9. React Native [Електронний ресурс] – Режим доступу: https://reactnative.dev/docs/getting–started Дата доступу: травень 2020.
10. HTML [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/ru/docs/Web/HTML Дата доступу: травень 2020.
11. CSS [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/ru/docs/Web/Guide/CSS/Getting_started/What_is_CSS Дата доступу: травень 2020.

12. JSX [Електронний ресурс] – Режим доступу: https://ru.reactjs.org/docs/introducing–jsx.html Дата доступу: травень 2020.

ДОДАТКИ

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2019 р.

ПРОГРАМНА ПЛАТФОРМА ДЛЯ ОБІКУ ВОЛОНТЕРСЬКОЇ РОБОТИ

Програма та методика тестування

ДП.045440-04-51

«ПОГОДЖЕНО"

Керівник проєкту:

_____ Анастасія ГРЕЧКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Нікіта МАСЛОВ

2019

ЗМІСТ

1. ОБ’ЄКТ ВИПРОБУВАНЬ

Платформа для обліку волонтерської роботи, який являє собою web-сайт та мобільній додаток, створений на платформі Node.js з використанням технології AJAX.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

1) функціональна працездатність елементів сторінок web-ресурсу;

2) наявність доступу до бази дистанційних курсів в системі EDU;

3) відповідність форматів та протоколів передачі даних з системою EDU;

4) забезпечення належного рівня безпеки даних;

5) зручність роботи з web-сайтом;

6) відповідність дизайну вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

Тестування виконується методом Black Box Testing. Перевіряється як код, так і безпосередньо програмний продукт на відповідність функціональним вимогам. Тестування відбувається на рівні «системного тестування».

Використовуються наступні методи:

1) функціональне тестування, зокрема на рівні Critical path test (базове тестування);
2) тестування продуктивності програмного забезпечення, зокрема Stability testing (тестування стабільності) та Load testing (навантажувальне тестування);
3) тестування інтерфейсу.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Тестування виконується засобами інструментарію SpecFlow.

Працездатність платформи перевіряється шляхом:

1) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
2) динамічного ручного тестування на відповідність функціональним вимогам;
3) статичного тестування коду;
4) тестування web-ресурсу в різних web-браузерах;
5) тестування при максимальному навантаженні;
6) тестування стабільності роботи при різних умовах;
7) тестування зручності використання;
8) тестування інтерфейсу.

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Науковий керівник кафедри

_____ Іван ДИЧКА

«___» _____ 2020 р.

ПРОГРАМНА ПЛАТФОРМА ДЛЯ ОБЛІКУ ВОЛОНТЕРСЬКОЇ РОБОТИ

Керівництво користувача

ДП.045440-05-34

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Анастасія ГРЕЧКО

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Нікіта МАСЛОВ

2020

ЗМІСТ

1. Опис структури платформи

Платформа для обліку волонтерської роботи складається з веб-додатку та мобільного додатку. Усі сторінки веб-додатку та мобільного додатку є динамічними.

До веб-додатку належать наступні сторінки:

- головна сторінка;
- сторінка перегляду подій;
- сторінка створення подій;
- сторінка перегляду студентів;
- сторінка перегляду статистики.

До мобільного додатку належать такі сторінки:

- головна сторінка;
- сторінка реєстрації;
- сторінка з подіями;
- сторінка з деталями події;
- сторінка створення відмітки часу;
- сторінка профілю волонтера.

Кожна сторінка веб-додатку, окрім головної, містить навігаційну панель, яка допомагає користувачу отримати доступ до інших сторінок веб-додатку, або завершити сесію користування та потрапити на головну сторінку.

Кожна сторінка мобільного додатку має знизу навігаційну панель, яка допомагає користувачу переміщуватися між сторінкою з подіями та сторінкою профілю волонтера.

2. Опис вмісту сторінок платформи

Головна сторінка містить форми для логіну та реєстрації директорів в системі. Форми змінюються за допомогою спеціальної кнопки без оновлення сторінки.

Форма логіну (рис. 1) пропоную користувачу ввести пару логін(електронна пошта) та пароль, щоб увійти у систему.

Рис. 1. Форма логіну

Форма реєстрації нового директора (рис. 2) містить поля для вводу електронної пошти, паролю та поле для підтвердження паролю. Після успішного проходження цього кроку, користувачу потрібно заповнити поля з інформацією по школі – повна назва навчального закладу та потрібну кількість годин для волонтера.

Рис. 2. Головна сторінка з формою реєстрації

Сторінка зі створеними подіями (рис. 3) містить список всіх створених директором подій. По кліку на подію, користувач отримує повну інформацію про подію(назву, адресу, опис, фото тощо).

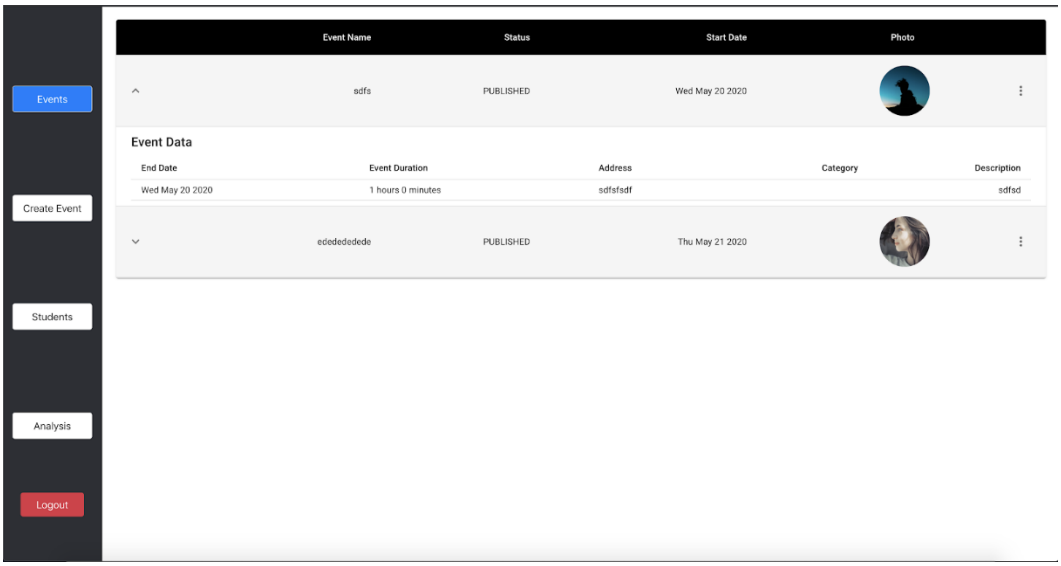


Рис. 3. Сторінка зі створеними подіями

Сторінка для створення подій (рис. 4) складається з форми з полями, яка дозволяє користувачу створити подію з потрібними полями. Користувач потрібен ввести ім’я події, її адресу, опис, вибрати фото, вказати дату, час та тривалість події. Якщо всі поля заповнені правильно, то буде створено подію,

яка буде відображена у мобільному додатку та на сторінки зі списком всіх подій веб-додатку.

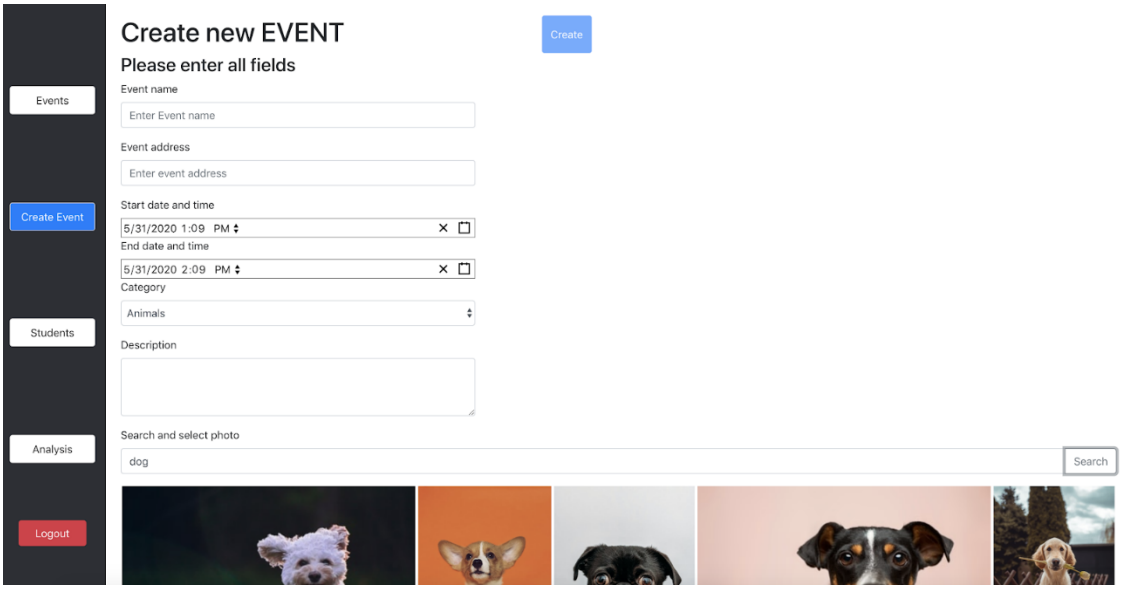


Рис. 4. Сторінка для створення подій

Сторінка з учнями в школі (рис. 5) складається зі списку всіх учнів, які відносяться до цієї школи. По кліку на учня, користувач отримує інформацію про учня та список його відміток. Кожна відмітка має 3 статуса: Pending, Rejected, Approve. Директор може змінювати статус відмітки.

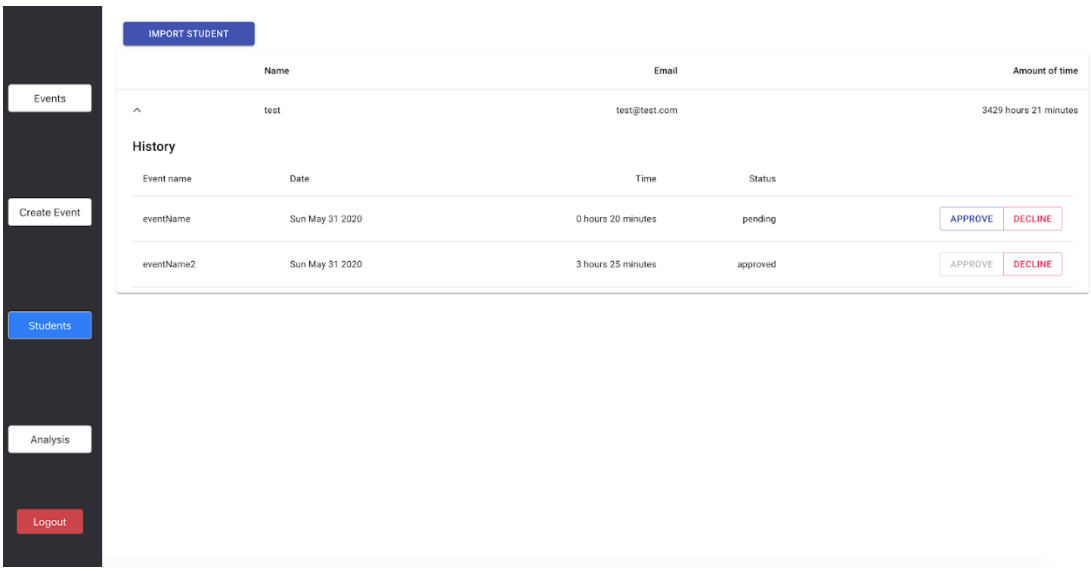


Рис. 5. Сторінка з учнями в школі

Головна сторінка містить форми для логіну та реєстрації волонтера в системі. Форми змінюються за допомогою спеціальної кнопки без оновлення сторінки.

Форма логіну (рис. 6) пропоную користувачу ввести пару логін(електронна пошта) та пароль, щоб увійти у систему. Якщо логін або пароль будуть введені неправильно, то буде показано повідомлення про помилку.

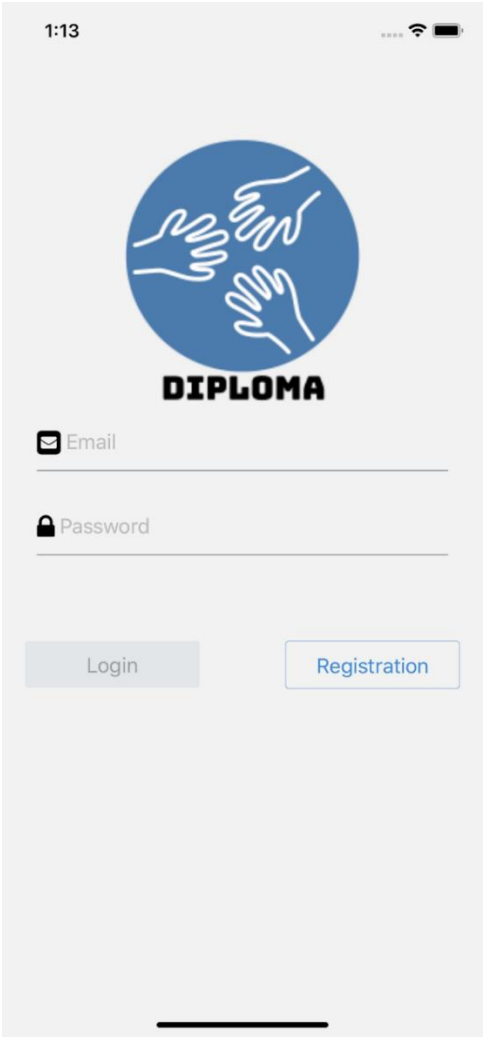


Рис. 6. Сторінка логіну

Сторінка реєстрації нового волонтера (рис. 7, 8) починається з того, що користувач має знайти та вибрати свою школу. Після цього учень потрапляє на сторінку реєстрації, де користувачу потрібно ввести електронну пошту, ім’я

та пароль. Якщо директор не створив даного учня, то користувач отримує помилку.

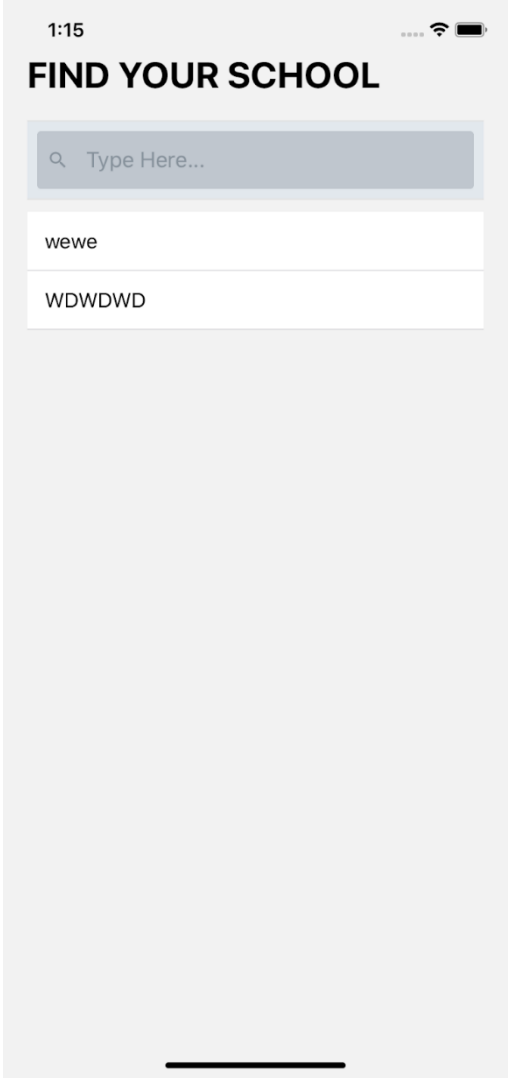


Рис. 7. Сторінка вибору школи

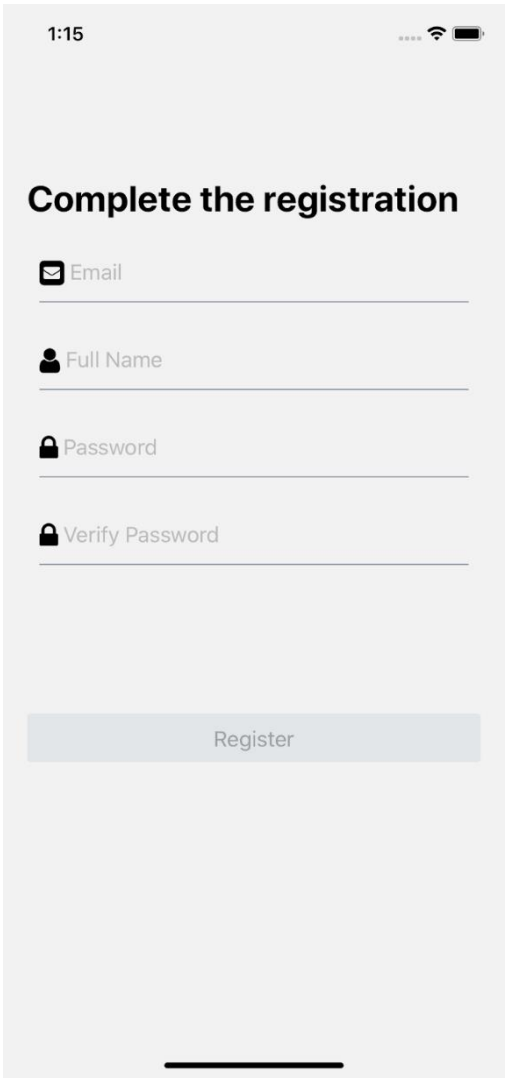


Рис. 8. Сторінка реєстрації

Сторінка з доступними подіями (рис. 9) складається зі списку доступних волонтеру подій. По кліку на подію користувач переходить до сторінки з інформацією про подію.

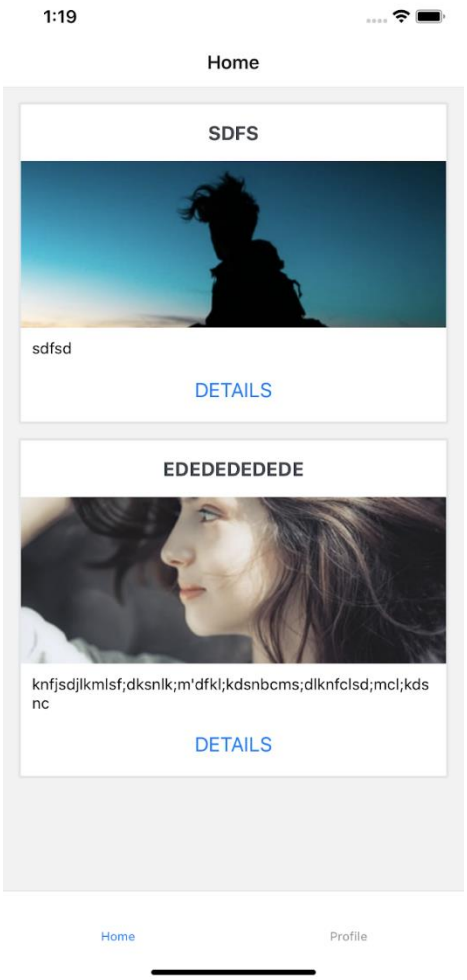


Рис. 9. Сторінка з доступними подіями

Сторінка з інформацією про подію (рис. 10) складається з інформації про подію та таймеру часу, який волонтер бере участь у події. Таймер має кнопку Start, яку неможливо натиснути до початку події. Після натискання, таймер почне рахувати час, а кнопка зміниться на Stop, після натискання якої таймер збереже час, та користувач перейде до сторінки створення відмітки.

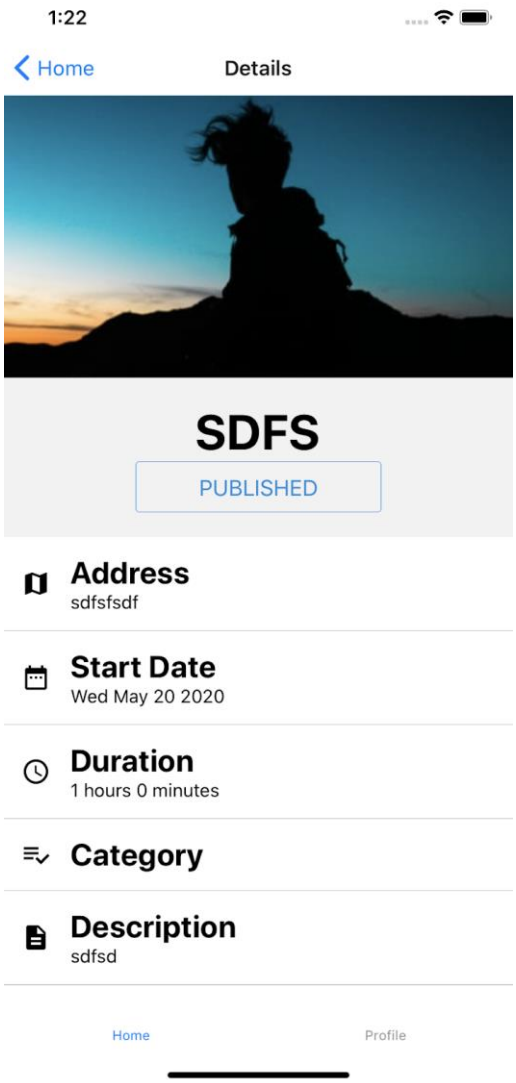


Рис. 10. Сторінка з інформацією про подію

Сторінка створення відмітки (рис. 11) складається з кількістю часу, яку рахував таймер, поле для вводу відгуку про подію, поле для вибору оцінки події, поле для вводу пароля відповідального, поле для оцінки волонтера та поле для написання електронної пошти відповідального за подію, щоб після створення відмітки, директор мав змогу перевірити правдивість відмітки.

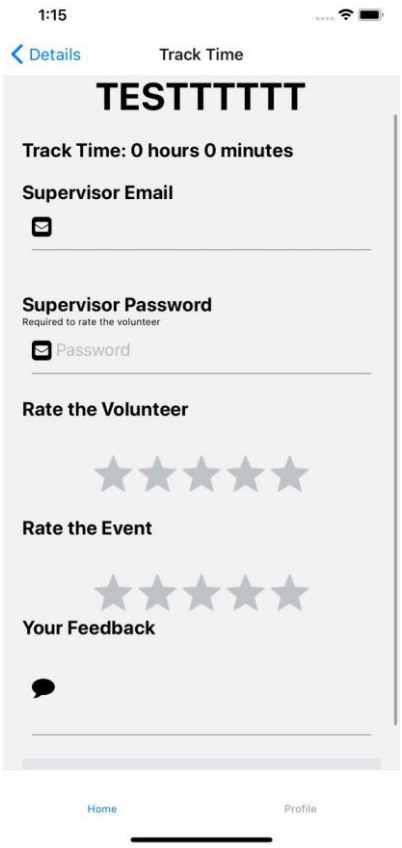


Рис. 11. Сторінка створення відмітки

Сторінка профілю волонтера (рис. 12) складається з інформації про користувача та його відміток, де волонтер має змогу слідкувати за статусом його відміток.

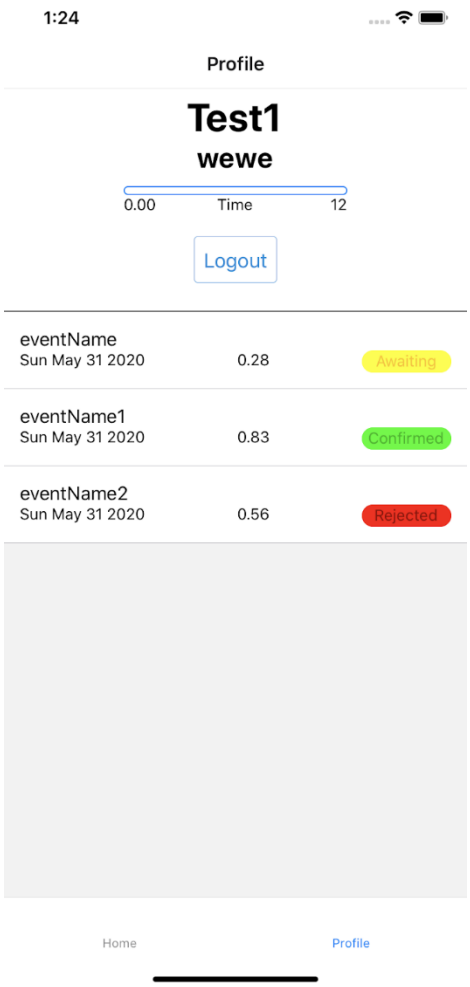


Рис. 12. Сторінка профілю волонтера

3. Процедура авторизації користувача

Авторизація користувача у веб-додатку відбувається на головній сторінці за допомогою форми для логіну (рис. 1). Користувач повинен ввести свій email та пароль, за допомогою яких користувач реєструвався. Якщо дані введені правильно, то користувач потрапляє до веб-додатку. Авторизація у мобільному додатку відбувається аналогічно, але у мобільному додатку (рис. 6).

4. Робота з подіями

Для початку, директор має створити подію на сторінці створення подій (рис. 4). Йому треба заповнити усі поля, і якщо все правильно, то створиться подія, яка буде доступна у веб-додатку (рис. 3) та мобільному додатку (рис. 9).

У веб додатку користувач має змогу змінити дані в події, якщо вона ще не почалася. Для цього треба вибрати подію, та у боковому меню, яке є для кожної події, треба вибрати «Update event». Після цього користувач потрапляє до сторінки створення подій, але поля будуть пре заповнені інформацією по події, яку користувач може змінити.

У тому же самому меню є ще кнопка для видалення події, яка носить назву «Delete event».

Якщо подія була створена та вона ще не почалася, то волонтер може побачити подію у мобільному додатку (рис. 9). Якщо він натисне на неї, то він потрапить до сторінки з деталями події (рис. 10). На цій сторінці буде кнопка, яка починає рахувати час після натискання на неї. Після цього кнопку можна натиснути ще раз, щоб потрапити на сторінку для створення відмітки про перебування на події (рис. 11).

5. Робота зі студентами

Директор повинен імпортувати студента до своєї школи. Це можна зробити на сторінці зі списком усіх студентів (рис. 5). На цій сторінці директор має змогу переглянути список усіх студентів та їх відміток.

Якщо директор не імпортував студента до школи, то студент не зможе пройти реєстрацію та потрапити до мобільного додатка. У кожного студента є потрібна кількість годин, які він має провести на волонтерських подіях. Інформацію про години волонтер може отримати на сторінці профілю волонтера (рис. 12).

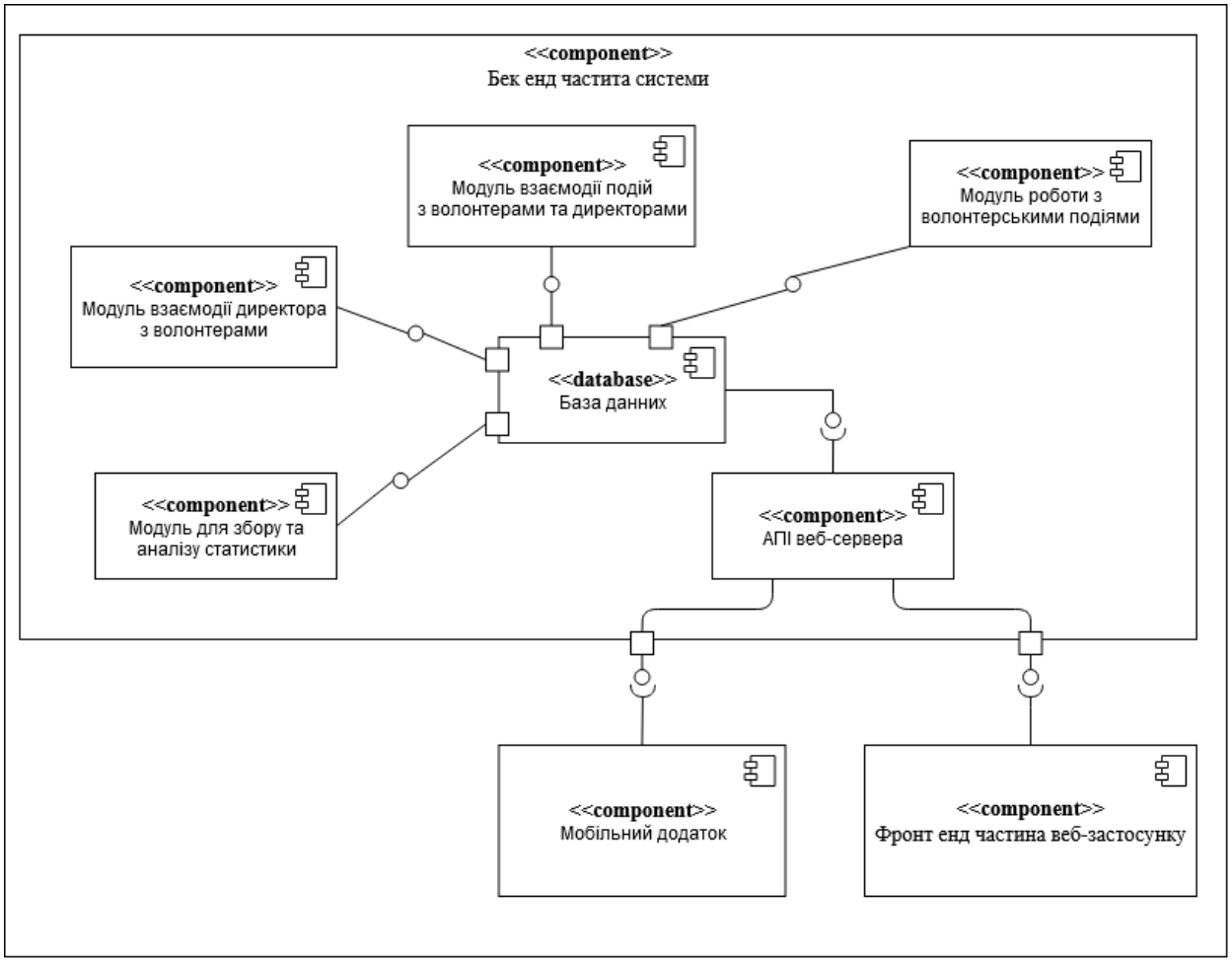
6. Робота з відмітками

При створенні відмітки, волонтер потрапляє на сторінку створення відмітки, де він має заповнити потрібні поля (email відповідального, відгук, оцінку події). Також там є спеціальні поля для відповідального, за допомогою яких він може поставити оцінку волонтеру. Він повинен ввести пароль, яких ввів директор при створенні події, і поставити оцінку волонтеру. Якщо пароль буде не вірній, то оцінка не зарахується.

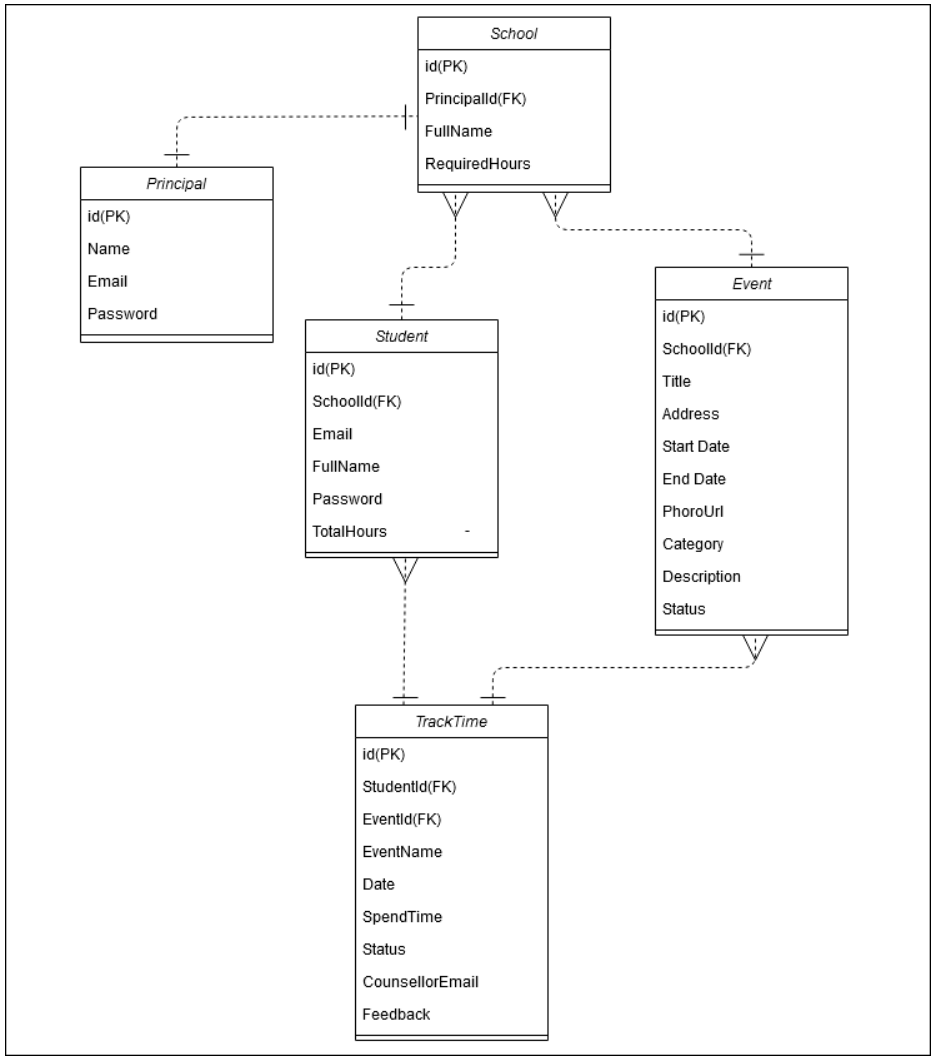
Коли волонтер створює відмітку про участь у події, то вона потрапляє до веб-додатку, де директор має змогу переглянути її, та змінити її статус. Якщо директор змінює статус на «Approve», то час, який є у відмітці, зараховується до часу волонтера, який він витратив на волонтерські події. Також директор може змінити статус на «Rejected», тоді час зараховуватися не буде. Статус відмітки можна змінювати у будь-який час.

Волонтер має змогу перевірити весь його зарахований час та статус усіх його відміток на сторінці профілю волонтера (рис. 12).

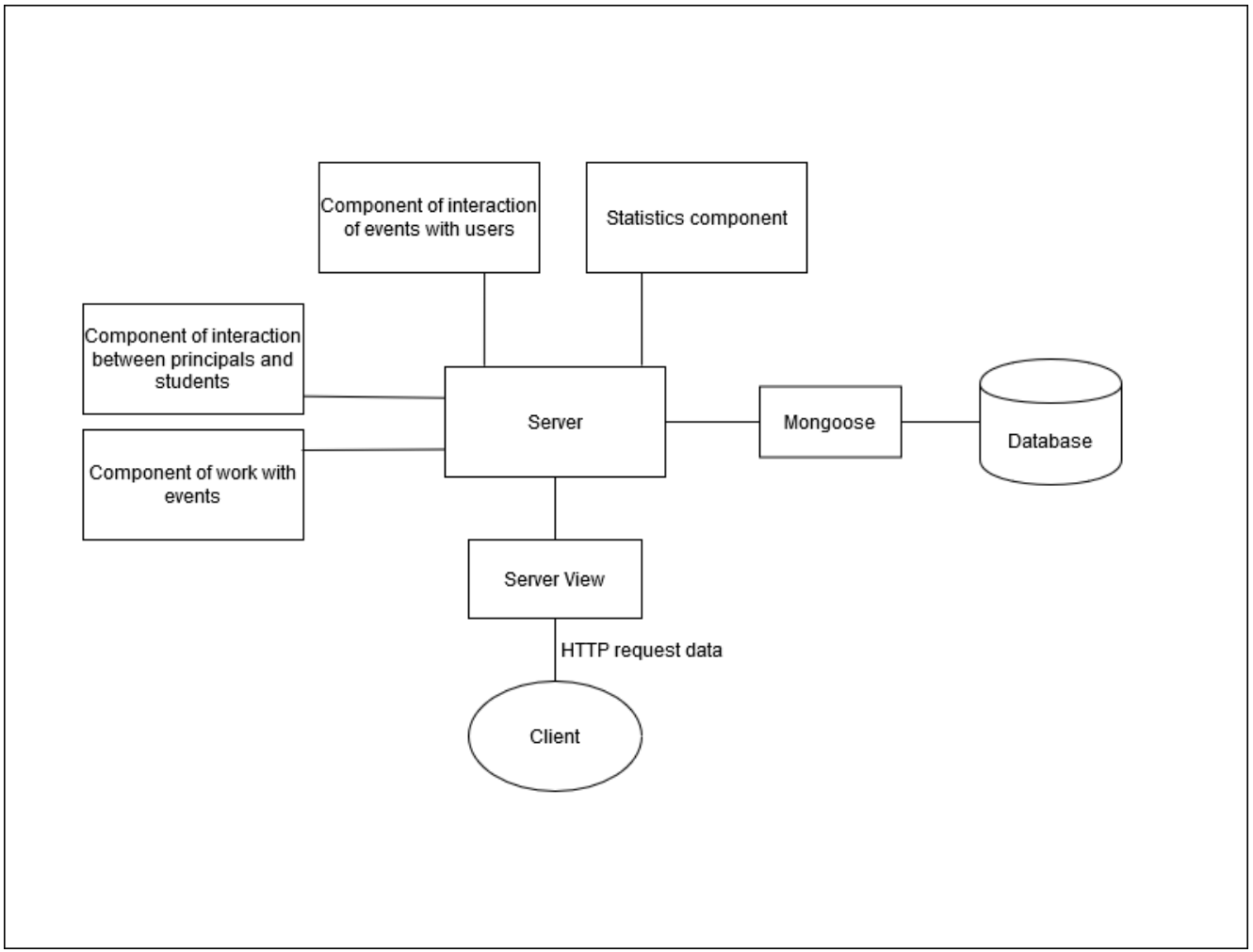
Додаток 1
Копії графічних матеріалів



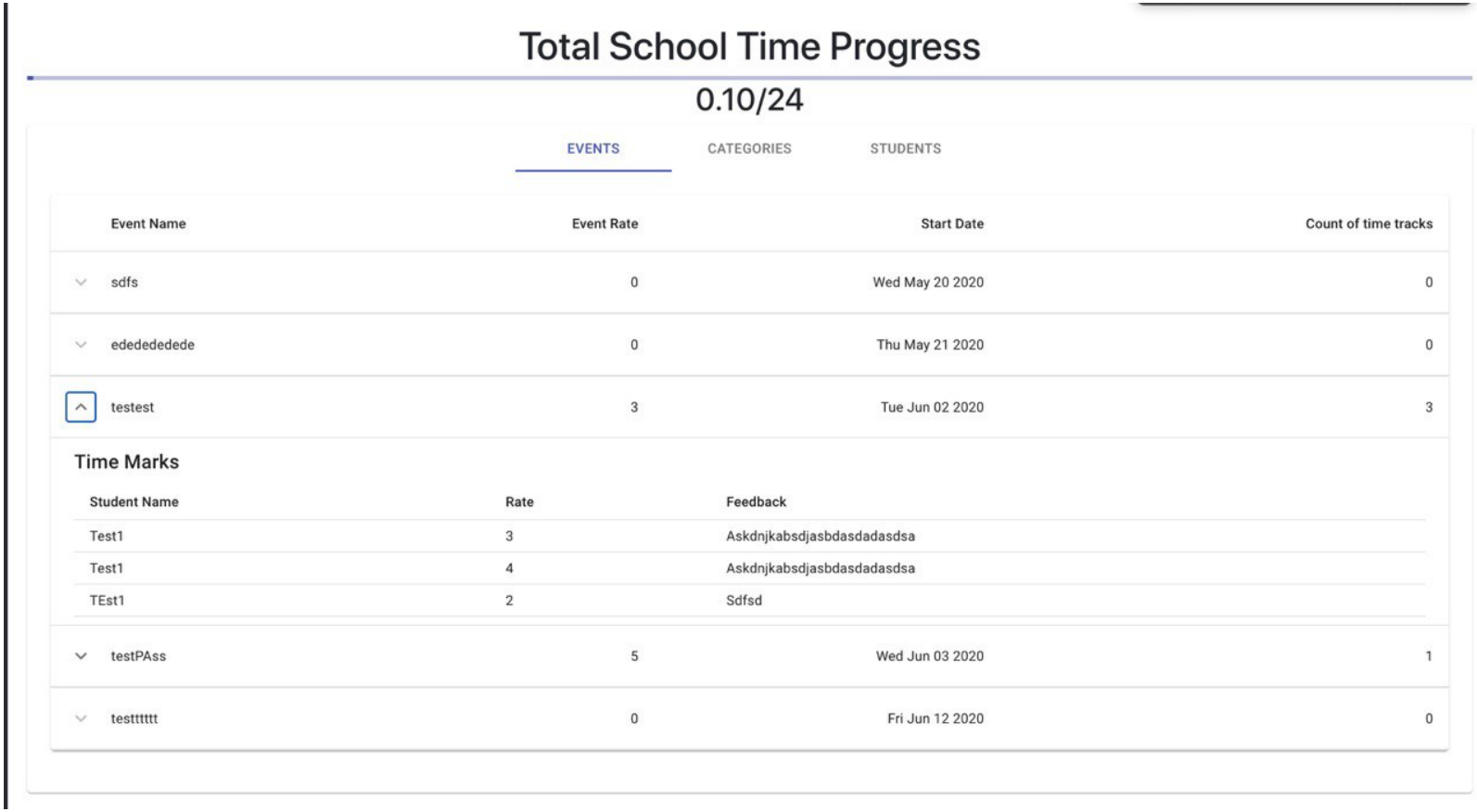
ДП.045440-06-99.
Платформа для обліку волонтерської роботи.
Архітектура системи. Компоненти платформи



ДП.045440-07-99.
Платформа для обліку волонтерської роботи.
Архітектура системи. Схема бази даних платформи. ER-діаграма



Маслов Нікіта Сергійович
гр. КП-62



Маслов Нікіта Сергійович
гр. КП-62

Додаток 2

Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП’ЮТЕРНИХ СИСТЕМ

ПЛАТФОРМА ДЛЯ ОБЛІКУ ВОЛОНТЕРСЬКОЇ РОБОТИ

Виконав: Н.С. Маслов

Науковий керівник: ст. викл. А.В. Гречко

Київ – 2020

АКТУАЛЬНІСТЬ

- Складний документообіг.
- Перехід до повної автоматизації обліку.
- Складність пошуку волонтерських робіт, де можуть підтвердити наявність волонтера.
- Можливе ухилення від робіт та підробка документів.
- Перевірка паперових звітів – монотонна задача.

ПОСТАНОВКА ЗАДАЧІ

Мета проекту: розробити програмну платформу для обліку волонтерської роботи учнів та студентів.

Завдання:

1. Проаналізувати існуючі рішення для обліку волонтерської роботи.
2. Розробити алгоритм роботи з волонтерськими подіями.
3. Розробити веб-додаток для директорів.
4. Розробити мобільний додаток для волонтерів.

АРХІТЕКТУРА СИСТЕМИ

- Клієнт-серверна взаємодія
- База даних
- Модуль роботи з волонтерськими подіями
- Модуль взаємодії директора з волонтерами
- Модуль взаємодії подій з волонтерами та директорами
- Модуль для збору статистики

ОБРАНІ ЗАСОБИ ВИРІШЕННЯ ПРОБЛЕМИ

Платформа для реалізації front-end.

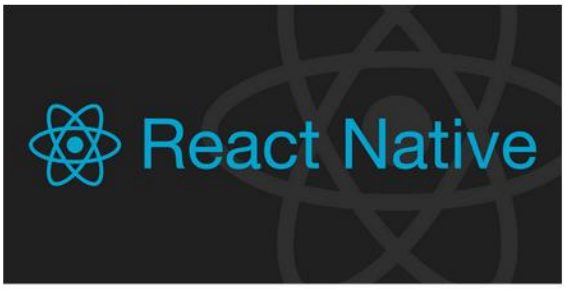
- Найбільша спільнота розробників серед платформ для front-end;
- Оптимізовані механізми для динамічного оновлення сторінок;
- Використання віртуального DOM-дерева.

ОБРАНІ ЗАСОБИ ВИРІШЕННЯ ПРОБЛЕМИ

Платформа для реалізації back-end.

- Велика спільнота розробників;
- Багато допоміжних модулей;
- Підтримка асинхронних запитів до сервера.

ОБРАНІ ЗАСОБИ ВИРІШЕННЯ ПРОБЛЕМИ



Платформа для реалізації mobile.

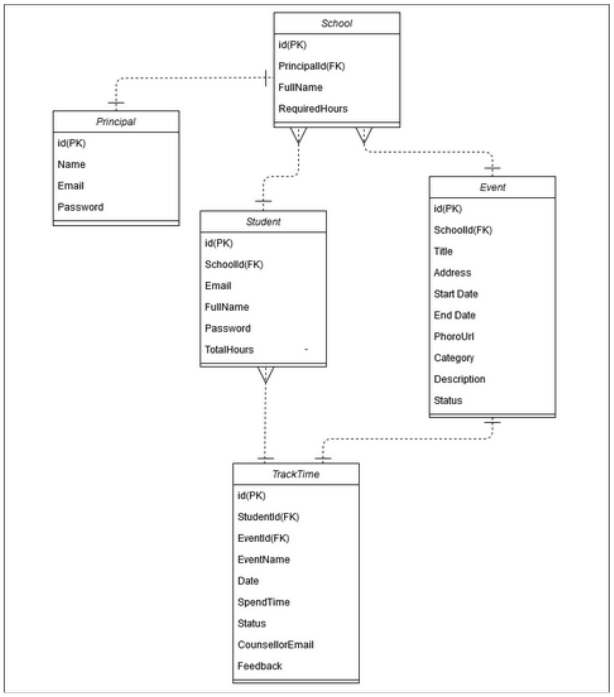
- Кросплатформеність;
- Підтримка JavaScript;
- Простота написання UI.

РОЗРОБЛЕНІ ЗАСОБИ. СТРУКТУРА БАЗИ ДАНИХ

СКБД – MongoDb

- Schools
- Principals
- Students
- Events
- TrackTimes

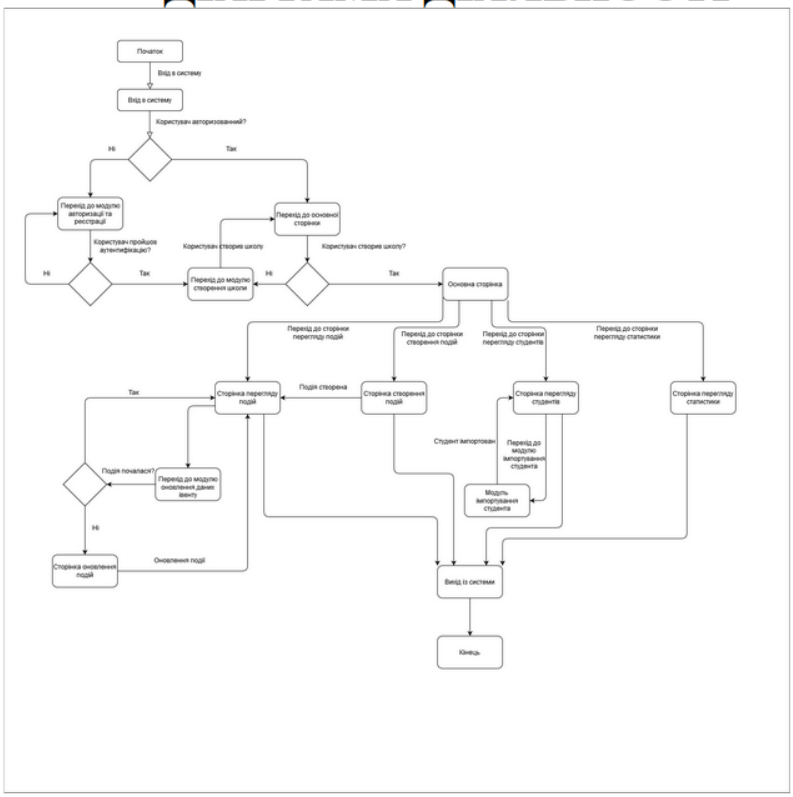
РОЗРОБЛЕНІ ЗАСОБИ. СТРУКТУРА БАЗИ ДАНИХ



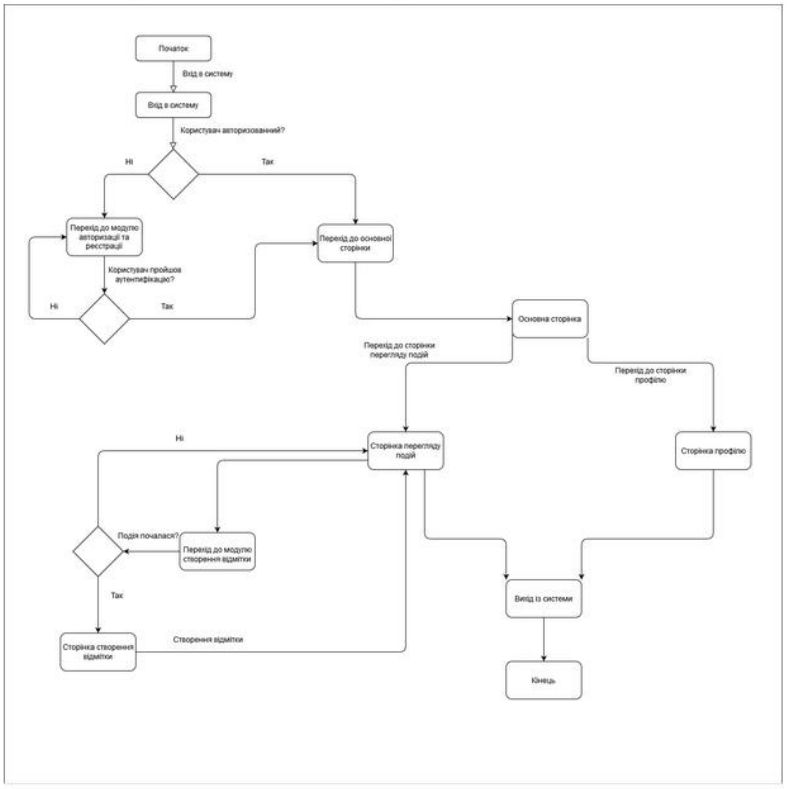
ОСНОВНІ ВИМОГИ ДО СИСТЕМИ

- Реєстрація та авторизація директорів;
- Імпорт студентів та подальша їх реєстрація;
- Створення волонтерських подій та відстеження їх статусу;
- Можливість студентів брати участь у події за допомогою мобільного додатка;
- Можливість студентів робити відмітки по участь;
- Можливість директорів змінювати статус відміток;
- Статистика по студентам, подіям та категоріям

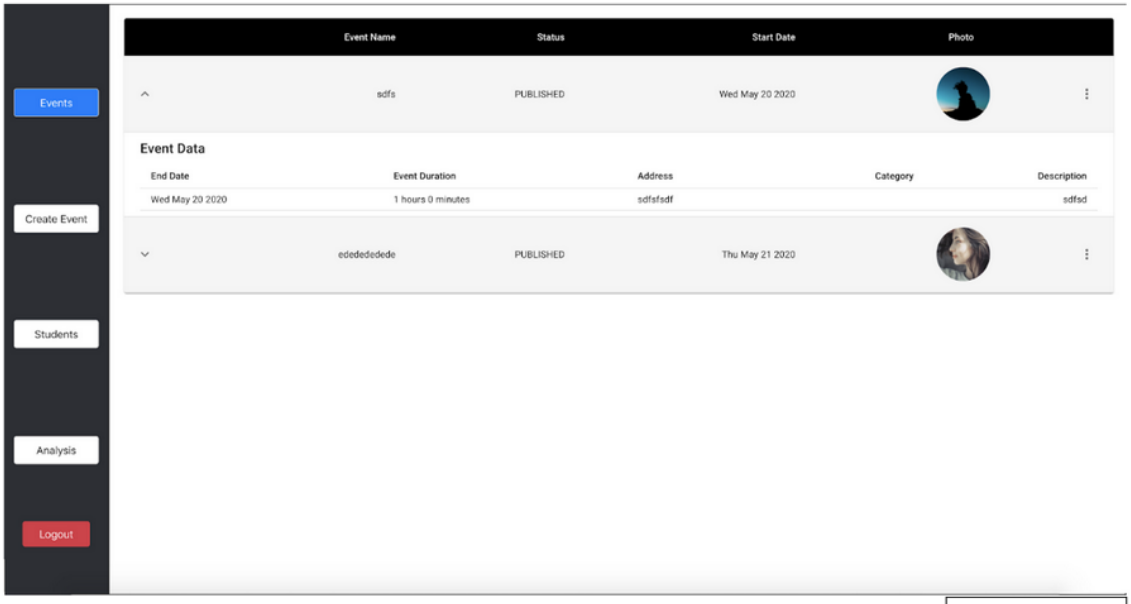
ДІАГРАМА ДІЯЛЬНОСТІ



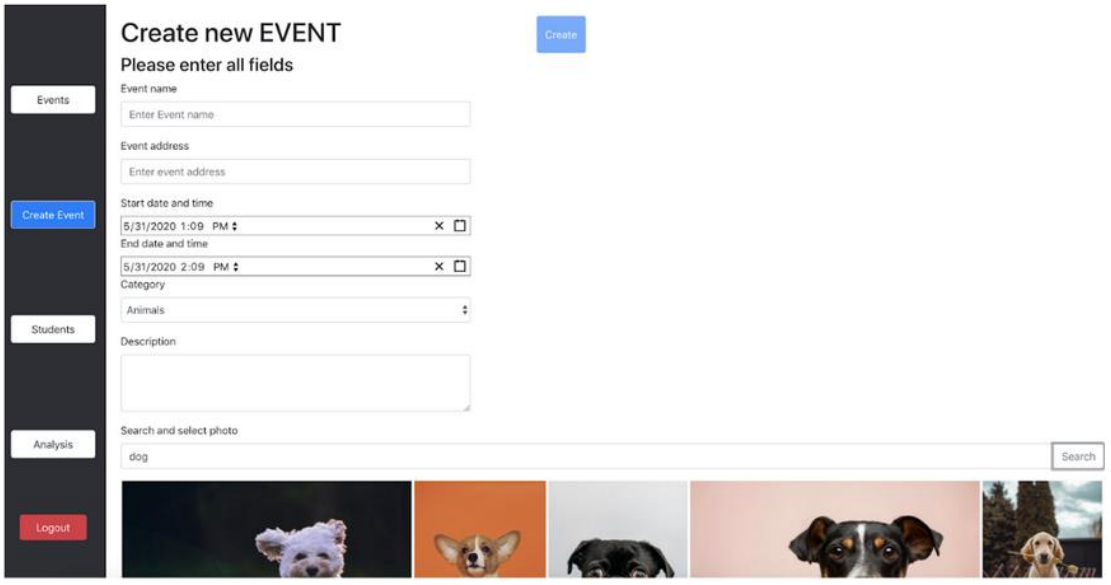
ДІАГРАМА ДІЯЛЬНОСТІ



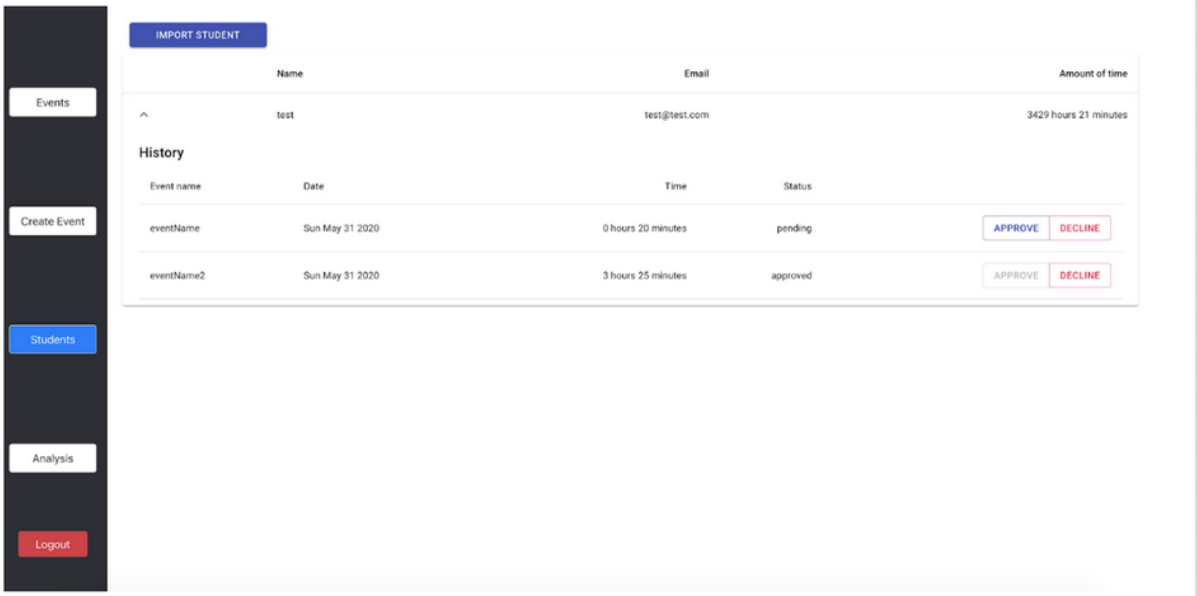
РОЗРОБЛЕНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ



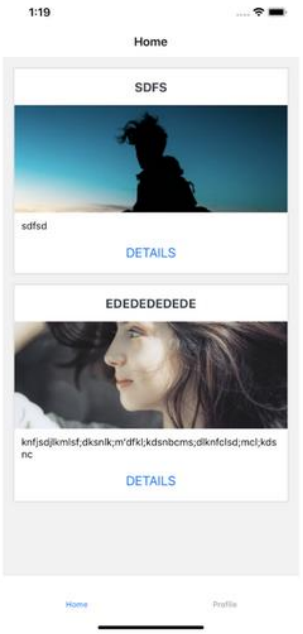
РОЗРОБЛЕНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ

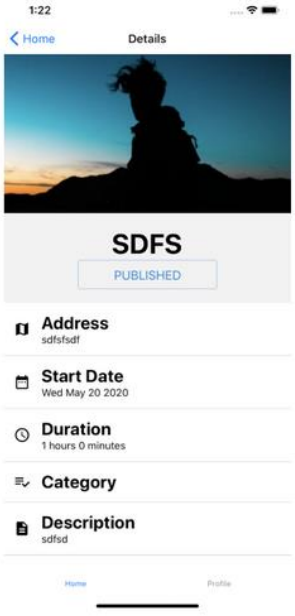


РОЗРОБЛЕНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ



РОЗРОБЛЕНЕ ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ





ВИСНОВКИ

1. Було проведено аналіз існуючих аналогів, виділено їх переваги та недоліки.
2. Розроблено клієнт-серверну архітектуру ПЗ.
3. Розроблено структуру бази даних.
4. Розроблено веб-додаток для роботи з системою для директорів.
5. Розроблено мобільний додаток для роботи з системою для директорів.
6. Протестовано програмні засоби.

Дякую за увагу!

Додаток 3

Текст програми

```
import React from 'react';
import PropTypes from 'prop-types';
import { makeStyles } from '@material-ui/core/styles';
import Table from '@material-ui/core/Table';
import TableBody from '@material-ui/core/TableBody';
import TableCell from '@material-ui/core/TableCell';
import TableContainer from '@material-ui/core/TableContainer';
import TableHead from '@material-ui/core/TableHead';
import TableRow from '@material-ui/core/TableRow';
import TableSortLabel from '@material-ui/core/TableSortLabel';
import Typography from '@material-ui/core/Typography';
import Paper from '@material-ui/core/Paper';
import IconButton from '@material-ui/core/IconButton';
import KeyboardArrowDownIcon from '@material-ui/icons/KeyboardArrowDown';
import KeyboardArrowUpIcon from '@material-ui/icons/KeyboardArrowUp';
import Collapse from "@material-ui/core/Collapse/Collapse";
import Box from "@material-ui/core/Box";


function descendingComparator(a, b, orderBy) {
    if (b[orderBy] < a[orderBy]) {
        return -1;
    }
    if (b[orderBy] > a[orderBy]) {
        return 1;
    }
    return 0;
}

function getComparator(order, orderBy) {
    return order === 'desc'
        ? (a, b) => descendingComparator(a, b, orderBy)
        : (a, b) => -descendingComparator(a, b, orderBy);
}

function stableSort(array, comparator) {
    const stabilizedThis = array.map((el, index) => [el, index]);
    stabilizedThis.sort((a, b) => {
        const order = comparator(a[0], b[0]);
        if (order !== 0) return order;
        return a[1] - b[1];
    });
    return stabilizedThis.map((el) => el[0]);
}

const headEventCells = [
    { id: 'name', numeric: false, disablePadding: true, label: 'Event Name'
},
    { id: 'rate', numeric: true, disablePadding: false, label: 'Event Rate'
},
    { id: 'start', numeric: true, disablePadding: false, label: 'Start
Date' },
    { id: 'marksCount', numeric: true, disablePadding: false, label: 'Count
of time tracks' },
];
const headCategoriesCells = [
    { id: 'name', numeric: false, disablePadding: true, label: 'Category
Name' },
    { id: 'rate', numeric: true, disablePadding: false, label: 'Category
Rate' },
    { id: 'eventsCount', numeric: true, disablePadding: false, label:
'Count of Category events' },
```

```
];
const headStudentCells = [
    { id: 'name', numeric: false, disablePadding: true, label: 'Student
Name' },
    { id: 'rate', numeric: true, disablePadding: false, label: 'Student
Rate' },
    { id: 'time', numeric: true, disablePadding: false, label: 'Student
Approved Time' },
    { id: 'marksCount', numeric: true, disablePadding: false, label: 'Count
of time tracks' },
];

function EnhancedTableHead(props) {
    const { classes, order, orderBy, onRequestSort, tableType } = props;
    const createSortHandler = (property) => (event) => {
        onRequestSort(event, property);
    };
    let headCells = [];
    switch (tableType) {
        case 'events':
            headCells = headEventCells;
            break;
        case 'categories':
            headCells = headCategoriesCells;
            break;
        case 'students':
            headCells = headStudentCells;
            break;
    }

    return (
        <TableHead>
            <TableRow>
                <TableCell padding="checkbox">
                </TableCell>
                {headCells.map((headCell) => (
                    <TableCell
                        key={headCell.id}
                        align={headCell.numeric ? 'right' : 'left'}
                        padding={headCell.disablePadding ? 'none' :
'default'}
                        sortDirection={orderBy === headCell.id ? order :
false}
                    >
                        <TableSortLabel
                            active={orderBy === headCell.id}
                            direction={orderBy === headCell.id ? order :
'asc'}
                            onClick={createSortHandler(headCell.id)}
                        >
                            {headCell.label}
                            {orderBy === headCell.id ? (
                                <span className={classes.visuallyHidden}>
                  {order === 'desc' ? 'sorted descending' : 'sorted
ascending'}
                </span>
                            ) : null}
                        </TableSortLabel>
                    </TableCell>
                ))}
            </TableRow>
        </TableHead>
    );
}
```

```
EnhancedTableHead.propTypes = {
    classes: PropTypes.object.isRequired,
    numSelected: PropTypes.number.isRequired,
    onRequestSort: PropTypes.func.isRequired,
    onSelectAllClick: PropTypes.func.isRequired,
    order: PropTypes.oneOf(['asc', 'desc']).isRequired,
    orderBy: PropTypes.string.isRequired,
    rowCount: PropTypes.number.isRequired,
};

const useStyles = makeStyles((theme) => ({
    root: {
        width: '100%',
    },
    paper: {
        width: '100%',
        marginBottom: theme.spacing(2),
    },
    table: {
        minWidth: 750,
    },
    visuallyHidden: {
        border: 0,
        clip: 'rect(0 0 0 0)',
        height: 1,
        margin: -1,
        overflow: 'hidden',
        padding: 0,
        position: 'absolute',
        top: 20,
        width: 1,
    },
}));


function EventTableBody({rows, isSelected, order, orderBy}) {
    const [open, setOpen] = React.useState({});
    return(
        <TableBody>
            {stableSort(rows, getComparator(order, orderBy))
                .map((row, index) => {
                    const isItemSelected = isSelected(row.name);
                    const labelId = `enhanced-table-checkbox-${index}`;

                    return (
                        <React.Fragment key={index}>
                            <TableRow
                                hover
                                aria-checked={isItemSelected}
                                tabIndex={-1}
                                key={row.name}
                                selected={isItemSelected}
                            >
                                <TableCell component="th" scope="row">
                                    <IconButton aria-label="expand row"
size="small" disabled={row.marksCount === 0} onClick={() => {
                                        let newOpen = {...open};
                                        newOpen[`${index}`] =
!newOpen[`${index}`];
                                        setOpen({...newOpen})
                                    }}>
                                        {open[`${index}`] ?
<KeyboardArrowUpIcon /> : <KeyboardArrowDownIcon />}
```

```
                                    </IconButton>
                                </TableCell>
                                <TableCell component="th" id={labelId}
scope="row" padding="none">
                                    {row.name}
                                </TableCell>
                                <TableCell
align="right">{row.rate}</TableCell>
                                <TableCell align="right">{new
Date(row.start).toDateString()}</TableCell>
                                <TableCell
align="right">{row.marksCount}</TableCell>
                            </TableRow>
                            <TableCell style={{ paddingBottom: 0,
paddingTop: 0 }} colSpan={6}>
                                <Collapse in={open[`${index}`]}
timeout="auto" unmountOnExit>
                                    <Box margin={1}>
                                        <Typography variant="h6"
gutterBottom component="div">
                                            Time Marks
                                        </Typography>
                                        <Table size="small" aria-
label="purchases">
                                            <TableHead>
                                                <TableRow>
                                                    <TableCell>Student
Name</TableCell>

<TableCell>Rate</TableCell>

<TableCell>Feedback</TableCell>
                                                </TableRow>
                                            </TableHead>
                                            <TableBody>
                                                {row.marks.map((item,
index)=>{

                                                    return(
                                                        <TableRow
key={index}>

                                                        <TableCell
component="th" scope="row">
                                                            {item.name}
                                                        </TableCell>


<TableCell>{item.rate}</TableCell>

<TableCell>{item.feedback}</TableCell>
                                                        </TableRow>
                                                    )
                                                    })}
                                            </TableBody>
                                        </Table>
                                    </Box>
                                </Collapse>
                            </TableCell>

                        </React.Fragment>
                    );
                })}
        </TableBody>
    )
}
```

```
function CategoryTableBody({rows, isSelected, order, orderBy}) {
    const [open, setOpen] = React.useState({});
    return(
        <TableBody>
            {stableSort(rows, getComparator(order, orderBy))
                .map((row, index) => {
                    const isItemSelected = isSelected(row.name);
                    const labelId = `enhanced-table-checkbox-${index}`;

                    return (
                        <React.Fragment key={index}>
                            <TableRow
                                hover
                                aria-checked={isItemSelected}
                                tabIndex={-1}
                                key={row.name}
                                selected={isItemSelected}
                            >
                                <TableCell component="th" scope="row">
                                    <IconButton aria-label="expand row"
size="small" disabled={row.eventsCount === 0} onClick={() => {
                                        let newOpen = {...open};
                                        newOpen[`${index}`] =
!newOpen[`${index}`];
                                        setOpen({...newOpen})
                                    }}>
                                        {open[`${index}`] ?
<KeyboardArrowUpIcon /> : <KeyboardArrowDownIcon />}
                                    </IconButton>
                                </TableCell>
                                <TableCell component="th" id={labelId}
scope="row" padding="none">
                                    {row.name}
                                </TableCell>
                                <TableCell
align="right">{row.rate}</TableCell>
                                <TableCell
align="right">{row.eventsCount}</TableCell>
                            </TableRow>
                            <TableCell style={{ paddingBottom: 0,
paddingTop: 0 }} colSpan={6}>
                                <Collapse in={open[`${index}`]}
timeout="auto" unmountOnExit>
                                    <Box margin={1}>
                                        <Typography variant="h6"
gutterBottom component="div">
                                            Events
                                        </Typography>
                                        <Table size="small" aria-
label="purchases">
                                            <TableHead>
                                                <TableRow>
                                                    <TableCell>Event
Name</TableCell>

<TableCell>Rate</TableCell>
                                                    <TableCell>Start
Date</TableCell>
                                                </TableRow>
                                            </TableHead>
                                            <TableBody>

{row.categoryEvents.map((item, index)=>{
                                                    return(
```

```
                                                                <TableRow
key={index}>
                                                                    <TableCell
component="th" scope="row">
                                                                        {item.name}
                                                                    </TableCell>

<TableCell>{item.rate}</TableCell>
                                                                    <TableCell>{new
Date(item.start).toDateString()}</TableCell>
                                                                </TableRow>
                                                            )
                                                        })}
                                                    </TableBody>
                                                </Table>
                                            </Box>
                                        </Collapse>
                                    </TableCell>

                                </React.Fragment>
                            );
                        })}
                </TableBody>
            )
}


function StudentTableBody({rows, isSelected, order, orderBy}) {
    const [open, setOpen] = React.useState({});
    return(
        <TableBody>
            {stableSort(rows, getComparator(order, orderBy))
                .map((row, index) => {
                    const isItemSelected = isSelected(row.name);
                    const labelId = `enhanced-table-checkbox-${index}`;

                    return (
                        <React.Fragment key={index}>
                            <TableRow
                                hover
                                aria-checked={isItemSelected}
                                tabIndex={-1}
                                key={row.name}
                                selected={isItemSelected}
                            >
                                <TableCell component="th" scope="row">
                                    <IconButton aria-label="expand row"
size="small" disabled={row.marksCount === 0} onClick={() => {
                                        let newOpen = {...open};
                                        newOpen[`${index}`] =
!newOpen[`${index}`];
                                        setOpen({...newOpen})
                                    }}>
                                        {open[`${index}`] ?
<KeyboardArrowUpIcon /> : <KeyboardArrowDownIcon />}
                                    </IconButton>
                                </TableCell>
                                <TableCell component="th" id={labelId}
scope="row" padding="none">
                                    {row.name}
                                </TableCell>
                                <TableCell
align="right">{row.rate}</TableCell>
```

```
                                <TableCell
align="right">{row.time}</TableCell>
                                <TableCell
align="right">{row.marksCount}</TableCell>
                            </TableRow>
                            <TableCell style={{ paddingBottom: 0,
paddingTop: 0 }} colSpan={6}>
                                <Collapse in={open[`${index}`]}
timeout="auto" unmountOnExit>
                                    <Box margin={1}>
                                        <Typography variant="h6"
gutterBottom component="div">
                                            Events
                                        </Typography>
                                        <Table size="small" aria-
label="purchases">
                                            <TableHead>
                                                <TableRow>
                                                    <TableCell>Event
Name</TableCell>

<TableCell>Rate</TableCell>

<TableCell>Time</TableCell>

<TableCell>Feedback</TableCell>

<TableCell>Status</TableCell>
                                                </TableRow>
                                            </TableHead>
                                            <TableBody>
                                                {row.marks.map((item,
index)=>{

                                                    return(
                                                        <TableRow
key={index}>
                                                            <TableCell
component="th" scope="row">

{item.eventName}
                                                            </TableCell>

<TableCell>{item.rate}</TableCell>

<TableCell>{item.time}</TableCell>

<TableCell>{item.feedback}</TableCell>

<TableCell>{item.status}</TableCell>
                                                        </TableRow>
                                                    )
                                                })}
                                            </TableBody>
                                        </Table>
                                    </Box>
                                </Collapse>
                            </TableCell>

                        </React.Fragment>
                    );
                })}
            </TableBody>
        )
}
```

```
export default function EnhancedTable({type, rows}) {
    const classes = useStyles();
    const [order, setOrder] = React.useState('asc');
    const [orderBy, setOrderBy] = React.useState('calories');
    const [selected, setSelected] = React.useState([]);

    const handleRequestSort = (event, property) => {
        const isAsc = orderBy === property && order === 'asc';
        setOrder(isAsc ? 'desc' : 'asc');
        setOrderBy(property);
    };

    const handleSelectAllClick = (event) => {
        if (event.target.checked) {
            const newSelecteds = rows.map((n) => n.name);
            setSelected(newSelecteds);
            return;
        }
        setSelected([]);
    };

    const isSelected = (name) => selected.indexOf(name) !== -1;

    let body = null;
    switch (type) {
        case 'events':
            body = (<EventTableBody rows={rows} isSelected={isSelected}
order={order} orderBy={orderBy}/>);
            break;
        case 'categories':
            body = (<CategoryTableBody rows={rows} isSelected={isSelected}
order={order} orderBy={orderBy}/>);
            break;
        case 'students':
            body = (<StudentTableBody rows={rows} isSelected={isSelected}
order={order} orderBy={orderBy}/>);
            break;

    }

    return (
        <div className={classes.root}>
            <Paper className={classes.paper}>
                <TableContainer>
                    <Table
                        className={classes.table}
                        aria-labelledby="tableTitle"
                        size={'medium'}
                        aria-label="enhanced table"
                    >
                        <EnhancedTableHead
                            tableType={type}
                            classes={classes}
                            numSelected={selected.length}
                            order={order}
                            orderBy={orderBy}
                            onSelectAllClick={handleSelectAllClick}
                            onRequestSort={handleRequestSort}
                            rowCount={rows.length}
                        />
                        {body}
                    </Table>
                </TableContainer>
```